ALMA Graphical User Interfaces

March 22, 2010

Authors

- Pierre Cubaud – CNAM (Conservatoire National des Arts et Métiers)
  cubaud@cnam.fr
- Emmanuel Pietriga – INRIA (Institut National de Recherche en Informatique et en Automatique)
  emmanuel.pietriga@inria.fr
- Alexandre Topol – CNAM (Conservatoire National des Arts et Métiers)
  topol@cnam.fr

Contents

1 Executive Summary **6**

2 Methodology **8**

3 Observing Tool **9**

3.1 General Comments 9

3.2 Project Structure 15

3.3 Spectral Tab 17

3.4 Spatial Tab 21

3.5 Miscellaneous 23

4 Executive Subsystem **25**

4.1 Overall Organization of the UI 25

4.2 ALMA Tab and Creation of Arrays 31

4.3 JLog 34

4.4 Alarm Panel 37

4.5 Control Devices 38

4.6 Scheduler, Project/SB Search, Dataflow 44

4.7 Quick Look 46

4.8 Miscellaneous 50

5 Control Room Layout **55**

List of Figures

1 Direct access to Preferences tabs in the Options window . . . 10
2 Reflect Phase in color scheme (green should be desaturated, blue is ok as is) . . . 11
3 Addressing the problem of limited screen real-estate . . . 12
4 Management of tabs . . . 13
5 Management of tabs . . . 13
6 Project Structure . . . 15
7 Target and Resources: the very dense layout impedes legibility . . . 16
8 Visualization of Spectral Lines - (a) current version, (b) dimming the absorption/transmission plot so that other components of the plot such as spectral lines better stand out. This might not be very convincing when printed on paper. Please refer to an on-line version. . . . 17
9 Visualization of Spectral Lines - transitions displayed with a light green color on a gray background are difficult to read . . . 18
10 Visualization of Spectral Lines - green bar moving in sync with the cursor . . . 19
11 Spectral Lines Selector: (a) matches transitions when putting parentheses, (b) does not if parentheses omitted . . . 20
12 Source Visualization . . . 21
13 Visualizing a FITS image in ds9, with a similar interface . . . 22
14 Group items in the toolbar according to their semantics, as in (b) . . . 23
15 Metadata associated with the proposal . . . 24
16 OMC interface with one array, showing the Scheduler plugin, the Mount Control plugin for one antenna, and the Dataflow plugin . . . 26
17 Basic examples of component layouts that help users build a stronger mental map of the system (Paranal/VLT). . . . 27
18 Multi-scale interface prototype with embedded device block diagram and baseline info . . . 28
19 Hierachical structure represented as a treemap: this visualization provides a compact yet legible view of the tree's components. . . . 30
20 JTree used for containers in ACS Overview panel . . . 31
21 Unconventional contract/expand icons . . . 32
22 Array tab management . . . 33
23 JLog running as a plugin . . . 34
24 Wrong use of a checkbox for mutually exclusive choice . . . 35
25 Alarm Panel running standalone . . . 37
26 Block diagram of the major devices in an antenna . . . 38

27 Diagrams for subsystems make it easier to find the information item that is searched for (Paranal/VLT) . . . 39
28 Different examples of schematic representations of devices with status information (Paranal/VLT) 40
29 Panel that would typically benefit from the user of a more visual representation of variables 41
30 Use of bar gauges to show values that are supposed to vary in specific ranges (Paranal/VLT) 41
31 . . . 42
32 Mount Status vs. Mount Control . . . 43
33 Project/SB Search panel, Scheduler panel in interactive array mode. . . . 44
34 Incorrect use of checkboxes for mutually-exclusive choices . . . 45
35 QuickLook running as an OMC plugin . . . 46
36 Problems of scalability in terms of number and complexity of the data plots to monitor . . 47
37 Problems of scalability in terms of number of antennas and baselines. Conventional charts and plots work well with a few antennas/baselines, but will not scale . . . 48
38 Part of the OMC status bar . . . 50
39 Highly-saturated colors are tiring for the human eye . . . 50
40 Blinking heart in the OMC interface (left) and at the VLT (right) . . . 51
41 Uncategorized views make for a log flat list that is tedious to scan . . . 51
42 Italic Fonts are legible on paper but perform poorly when displayed on screen . . . 52
43 Excessive nesting of components makes it difficult for users to orient themselves in the interface . . . 53
44 Web browser plugin for the OMC . . . 54

1 Executive Summary

This document provides feedback about the current status of ALMA GUIs and suggestions for improvements. It covers the Observation Tool and various Executive Subsystem components, including the Operator Monitoring and Control framework, JLog and the Alarm Panel, Control Device GUIs and QuickLook.

Observation Tool: The target audience of this application is the community of radio-astronomers who will use it to prepare their observation proposals. It is important for the OT to be easy to learn and use. We provide suggestions for enhancements. These, however, do not completely address the main issue: the current GUI structure prevents novice users from building a clear mental model of the application and observation proposal editing process. Users are prone to getting lost in the large set of panels and components that constitute the OT. Addressing this issue would require to spend more time with both designers and end-users of the OT. Two general approaches to address this issue are proposed: 1) try to linearize the observation proposal's input process by re-considering the current GUI structure; 2) keep the current structure, but make the contextual help even more contextual with finer grained suggestions about possible next steps.

Executive Subsystem: From a usability perspective, the **OMC** is very different from the OT, in the sense that it will be used by a small number of expert users that will be exposed to it for long periods of time, daily. Power users have the capability to adapt to complex interfaces, if only because their job requires it, but bad UI design can have strong negative consequences even for this type of user. A difficult-to-use, over-crowded UI will be the source of more errors; it also means that users are more likely to fail to react quickly enough to a problem, simply because they are cognitively overwhelmed with data and because their interaction with the software is impeded by tedious UI navigation tasks.

The current interface based on a WIMP docking framework does not scale to the large number of plugins and antennas that have to be dealt with. Because of the number of panels and antennas, users will spend a lot of time managing windows, as only a very limited subset of all information can fit on screen, even when using three or four displays per workstation. This will seriously impede navigation in the interface, and consequently operator/astronomer efficiency. Furthermore, this very abstract presentation of the information prevents users from building a clear mental map of the system, which would allow them to navigate in the interface more efficiently, leveraging low-level cognitive abilities such as spatial memory and spatial orientation. Interviews with astronomers also revealed that important information is currently difficult to put in context, e.g., with respect to baseline length or meteorological conditions, making the user's task more cognitively demanding.

We provide concrete suggestions for enhancements, but these only partially address the main issue described above. This issue represents a major UI challenge that cannot be addressed by incremental enhancements. The overall organization of the UI should be reconsidered, in favor of more advanced information presentation and navigation paradigms such as, e.g., multi-scale user interfaces [1, 9, 16, 23] and multiple coordinated views [22], that will enable a more efficient use of available screen real-estate and will let users navigate more efficiently in the information space. This does not require completely reimplementing the OMC. Many of the plugin panels can be kept almost "as is".

As far as **QuickLook** is concerned, the current plotting solutions will hardly scale to 60+ antennas, and will definitely not when thousands of baselines have to be represented. Making QuickLook scale requires more UI design work, working closely with end users (operators and astronomers) to understand how these plots will be used (What information is critical? How often do users look at them? What actions

are they likely to take in response to problems identified in the plots? What is the time frame/delay before these actions get reflected in the plots, if at all? Do users need to establish correlations between events in different plots? etc.). Sketches of possible solutions are provided in this document. Easier-to-implement suggestions for improvements to the current UI are also given. However, these are clearly incremental and do not address the above issues on their own.

Conclusion: Our general impression is that GUIs were given relatively little attention, and that several decisions that impact them were made from a software engineering perspective, ignoring the actual needs of users. One example of this is the fact that plugins seem to be totally independent from one another. While this makes for an elegant architecture, it also means that plugins cannot communicate with one another; consequently the corresponding UI components cannot easily be synchronized, though this would often make sense (see, e.g., Sections 4.1, 4.3, 4.4).

For critical and complex systems such as ALMA, careful UI design is very important, and given the complexity of the GUIs, this work should be conducted by people who are knowledgeable in terms of UI design and who are well-aware of the state of the art in terms of human computer interaction and information visualization. Designing UIs for complex systems actually requires expertise, that software engineers often do not have, simply because they have not been trained for this. To be blunt, being able to programmatically instantiate UI components and assemble them in complex layouts is not sufficient to create efficient GUIs; as being able to instantiate multiple threads is not sufficient to create an efficient and deadlock-free multi-threaded architecture.

Scalability to 60+ antennas and more than two thousand baselines will be difficult to achieve with the current interface design. A multi-scale interface embedding enhanced versions of existing plugins, along with multiple coordinated views, would represent a major step forward and would be much more scalable. Designing and implementing such a solution requires significant work: participatory design workshops with operators and astronomers; design of appropriate visualizations and associated interactions; selecting, learning and using multi-scale structured graphics and information visualization toolkits for their implementation; integration of plugins, including support for communication between some of them.

2 Methodology

We conducted interviews with ALMA personnel, including operators, astronomers and software developers in December 2009, both in the Santiago offices, at the OSF, and at the VLT in Paranal. We later explored the Observing Tool's interface by installing it locally on our own computers. Emmanuel Pietriga was granted access to an STE instance to explore the OMC's UI through a remote connection for half a day in February. The following documents served as reference material for our report: [4, 10, 15, 20, 27, 28, 29], as well as screenshots from various systems both for ALMA and the VLT.

The following people were interviewed:

- Emilio Barrios (OSF)
- Stuart Corder (Santiago)
- Angela Cortes (Paranal/VLT)
- Bill Dent (OSF)
- Philippe Duhoux (Paranal/VLT)
- Alessandro Caproni (Santiago and OSF)
- Preben Grosbøl (OSF)
- Antonio S. Hales (OSF)
- Valentino Ivanov (Paranal/VLT)
- Alison Peck (Santiago)
- Mark Rawlings (OSF)
- Kartik Sheth (OSF)
- Debra Shepherd (OSF)
- Baltazar Vila-Vilaro (Santiago and OSF)
- Nick Whyborn (OSF)
- VLT-I operators (Paranal/VLT)

ALMA GUIs were also discussed extensively with Joseph Schwarz throughout this period.

3 Observing Tool

Contact Author: emmanuel.pietriga@inria.fr

Syntax used for suggestions: **S(difficulty, impact):** ...

3.1 General Comments

The OT can be seen as a domain-specific structured document editor. The interface is implemented with conventional WIMP [7] widgets in Swing. The primary elements of the interface are separated using resizable split panes. This allows users to easily adapt the UI layout based on available screen real-estate. **S(easy, low):** Ideally, the application would remember the relative size of the respective split panes. Right now it looks like the split panes' dimensions get initialized to their default value when the application starts, instead of restoring the panes' dimensions according to the user's configuration from the last session. Impact is low here because there aren't that many split panes overall (4 currently).

Menu items, toolbar buttons and other widgets get disabled when they cannot be used, which is good considering the number of UI components. I would however suggest that the number of UI components can be decreased in some places without losing any feature. For instance, all tabs of the `Options` window can be accessed directly from the `Options` menu (Figure 1). It is nice to be able to access any of the Options tab directly, but 1), this is unusual, and 2) this is not essential as users will seldom open this window. It seems more important to decrease the number of UI components than to enable fast access to this part of the interface. **S(easy, moderate):** The `Options` menu could be removed, replaced by a single menu item (called `Preferences` rather than `Options`) in either the `Edit` menu or `File` menu. That menu item would trigger the `Options` window, either always on the first tab (easier), or the last selected tab (better, but not essential).

The `Overview` pane contains useful information about the different high-level steps one should take. Some more context-dependent help is directly integrated in the `Forms` tab. The `Feedback` pane provides incremental feedback about the validity of the proposal and can take the user to the location in the interface where he can address issues. I haven't had the opportunity to investigate this feature deep enough to say whether it does a good job or a perfect job, but this is clearly a very important feature that deserves to be as rich as possible. One minor comment: it is not obvious that clicking on a line in the `Feedback` section sometimes takes you to/highlights the source of the problem. **S(easy, moderate):** add an icon such as an arrow that would symbolize the action of *going to* the source of the error (both the line and the icon should be clickable).

The primary navigation facility is the JTree on the left-hand side, with tabs in the other panes to navigate between different sections of the document. There are several issues associated with this navigation model. It is not linear, meaning that newbies are lost. There is no obvious place where to start, and where to go next after having filled-in one section. It is relatively unclear when you are done with the observation proposal's editing process. I am not claiming here that the specification process can be completely linearized, but trying to better guide the user w.r.t where he should start and where he could/should go next after each step, at a fairly fine grain, would have a significant positive impact on the overall user experience. Re-thinking how information is presented in terms of structure (which has a direct impact on how the user navigates in the interface) would be another, complementary way of addressing this issue. The mental

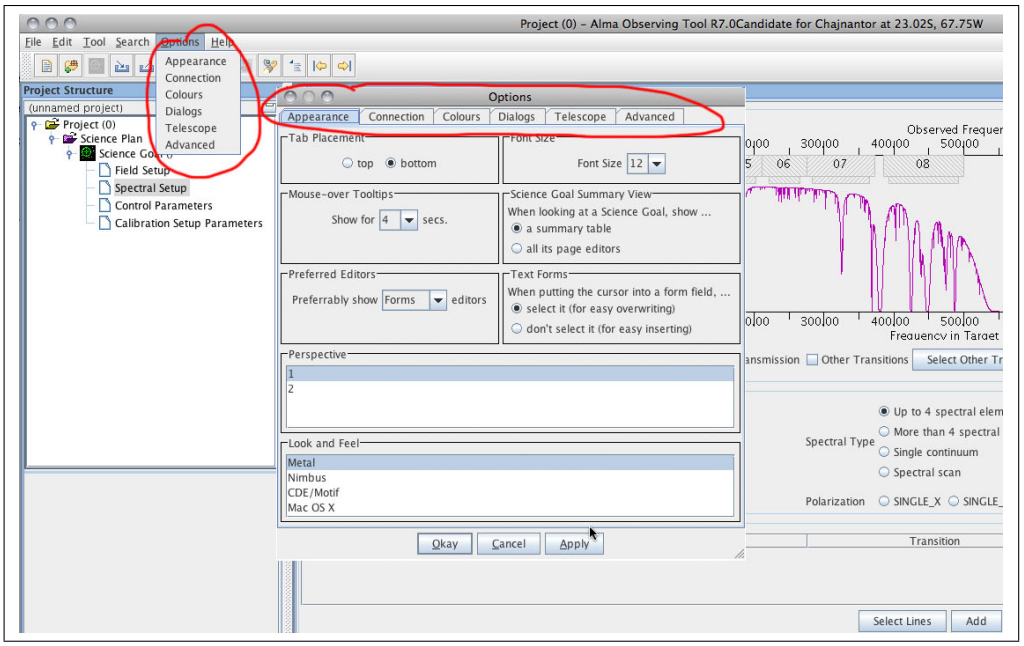


Figure 1: Direct access to Preferences tabs in the Options window

model that the user has about the tool and the specification process is very important. It is extremely frustrating to have very constrained editors that don't let you specify things according to your own mental model of the task, but here it is the opposite: you have very few constraints and are not guided much. This does not help build a clear mental model of the application. Addressing this means re-considering (not necessarily discarding, but at least modifying) the JTree+tabbed panes GUI structure. **S(hard, major):** what is the best solution is not entirely clear, and depends on the users' mental model of the task. Though we interviewed a few astronomers about the OT in Chile, we do not have enough data about this to propose a full-fledged solution to the problem. The OT offers templates for common observations, that can be edited by users. This is a good start, though I do not think it will be sufficient. Addressing this issue would require more time with both designers and end-users of the OT. As a summary, two approaches to solving this issue are: 1) try to linearize the obs. proposal's editing process by re-considering the JTree+tabbed panes GUI structure (allowing users to go back to the previous steps), thus enforcing one particular path to the obs. proposal's specification; 2) keep the current structure, but make the contextual help even more contextual, suggesting next steps at a finer-grain than what is currently done. The earlier-mentioned pre-filled observation templates would be good additions to any of these two solutions.

Other comments related to the general organization of the GUI follow.

The current grouping of components in the Spectral/Spatial/Forms/Catalog tabs is the source of several problems beyond the one mentioned above. First, there is a problem of consistency in the interface: the content of the `Forms` tab changes radically depending on what item is selected in the JTree, whereas the content of other tabs (`Spectral, Spatial`) does not. This inconsistency is misleading: if the `Forms` tab's content varies depending on the currently-selected JTree item, then one expects that the content of all other tabs will also depend on the currently-selected JTree item, because they all belong to the same tab

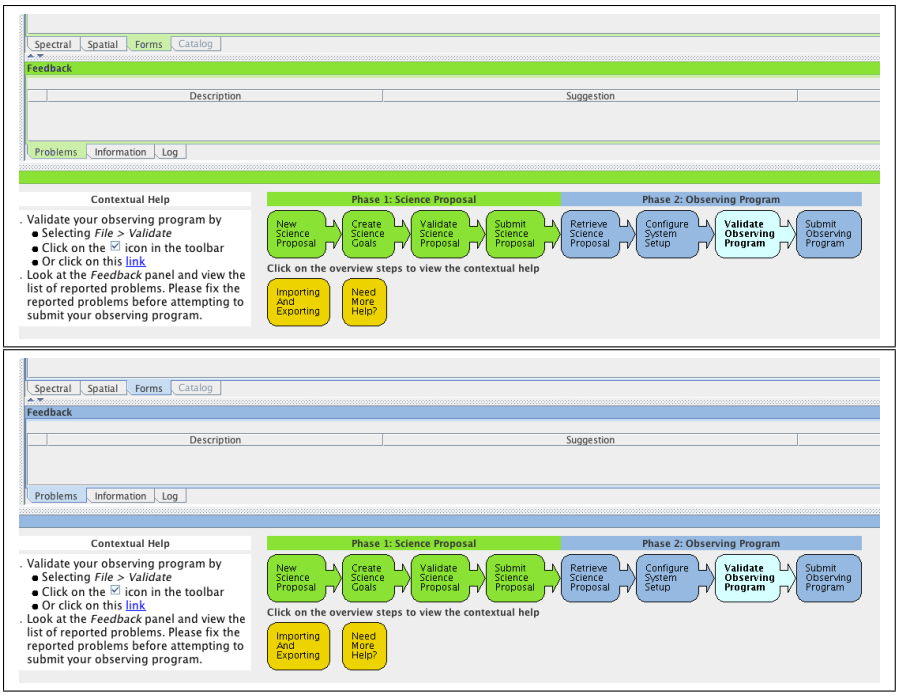


Figure 2: Reflect Phase in color scheme (green should be desaturated, blue is ok as is)

group.

S(easy, major): reorganize the tabs based on their relationship to the JTree. Actually, this is a related to the high-level suggestion above. Note that this particular issue of consistency can be solved locally, even if you do not re-consider the overall UI organization.

S(easy, moderate): the generic tab name `Forms` should be changed to something more informative, maybe context-sensitive, i.e., depending on what is selected in the JTree.

A somewhat related issue exists with the `Proposal` and `Program` tabs. These more or less correspond to the main two phases in the Overview pane. But when one switches from one tab to the other, very few things change. I understand that there is a strong intersection between the two phases, if only because the observing program heavily depends on what has been specified in the science proposal, but the interface fails to provide the user with any contextual information about this. Scheduling Blocks (SBs) disappear when going to the proposal tab, but most of the remaining parts of the interface do not change. It is good in a way since from Program you can do everything you can do from Proposal, but still, it is confusing. **S(easy, moderate):** the two phases are color-coded in the `Overview`. Why not use different color schemes for the widgets in the interface depending on whether the user is in `Proposal` mode or `Program` mode? For instance, the background color of the title bar of each split pane (`Project Structure, Editors, ...`) could change depending on which of the two modes is selected. Same thing for the 'focus color' (currently selected tabs). See Figure 2. The Program blue color is nice. The Proposal green color should be made less saturated if it is to be used in more places in the GUI as suggested here.

General issue with widget organization in the `Editors` pane: one doesn't always realize that there is

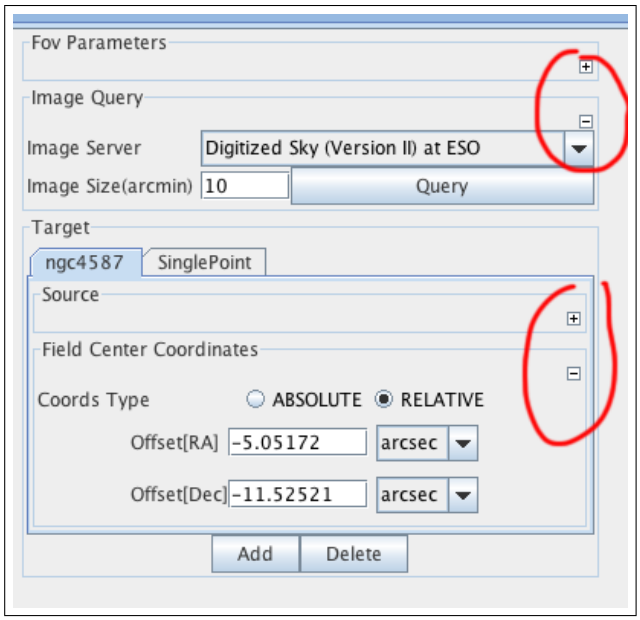


Figure 3: Addressing the problem of limited screen real-estate

more content than what is displayed at first in this pane. In other words, users don't always see they have to scroll down, that there are more fields below, simply because the nesting and quantity of UI components, scrollbars, etc., hides it. Another issue is that of mixing two different techniques for addressing the problem of limited screen real-estate (panes that don't fully fit on screen): one consists in adding scroll bars; the other in making it possible for the user to contract/expand sections of a pane with small +/- boxes (Figure 3). These two techniques aren't really compatible, and one should not mix them. **S(easy, moderate):** stick to one, possibly enhancing it to make navigation from section to section easier. Remove the collapse/expand feature. This means that everything has to be expanded. If this generates very long scrollable panels, you could add a menu whose items are the different sections of the pane (like `Fov parameters, Image Query, Target` in Figure 3). Clicking on one of these items would scroll the viewport automatically to that section. This would be similar to anchor-based navigation in Web pages (intra links in a given page).

On the location of some widgets: some buttons are located too far from what they act on. Figure 4 illustrates one such issue: the `Add` and `Delete` buttons are actually used to manage the tabs (adding a new tab, deleting the selected one). But their semantics is ambiguous. Being at the bottom of the form, one does not think that they are related to tab management. It looks more like you have to click on `Add` to submit/validate the source that you just input in the form above (which is not the case). **S(easy, high):** at least rename `Add` and `Delete` to things like `Create Additional Source` and `Delete Current Source`; better: replace `Delete` with small crosses at the right of each tab label, as is done in other parts of the interface (see, e.g., Figure 5) and in many tab-based interfaces such as Web browsers (this will also make tab management more consistent throughout the OT); move the renamed `Add` button either to the top, next to the tab labels, or keep it at the bottom, but move it to the far left or far right. Its meaning will be less ambiguous that way. Better: replace the Add button by a + icon on the right side of the rightmost tab, as is done in the latest versions of Firefox or Safari (the latter puts the icon on the far

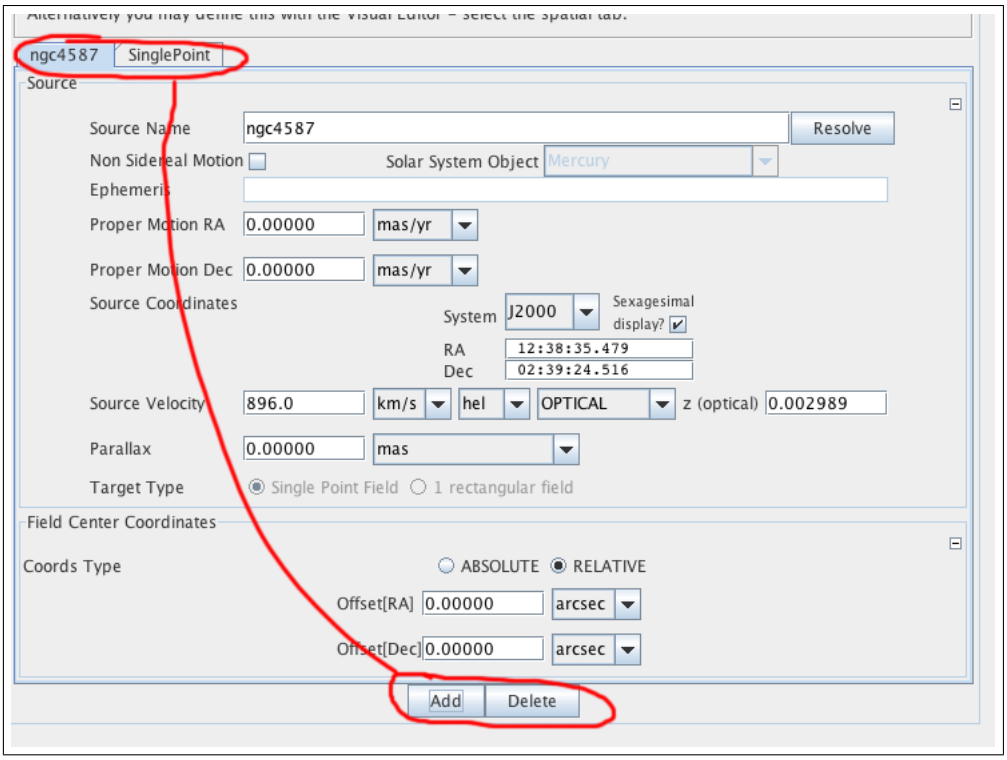


Figure 4: Management of tabs

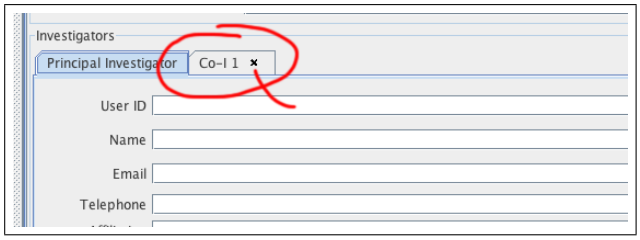


Figure 5: Management of tabs

right, which seems less convenient).

3.2 Project Structure

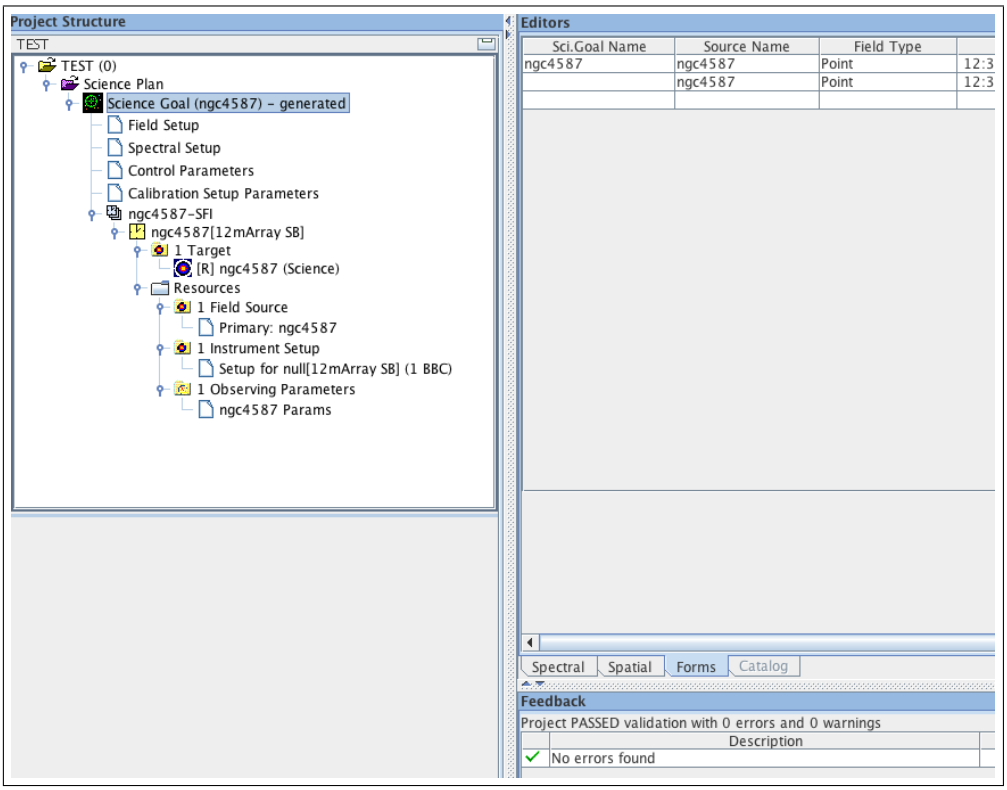


Figure 6: Project Structure

In the JTree, scheduling blocks generated from a given science goal are presented as children of the science goal node (Figure 6). But apparently there can only be one top-level SB node (which, I guess, corresponds to a ObsUnitSet) in the JTree for a set of coherent SBs (trying to *Generate SBs from the Selected Goal*, when this has been done before, prompts the user for a choice: overwriting the previously generated ObsUnitSet, or cancelling). From what we were told, users would like to use the same science goal to generate multiple top-level SBs, concrete observations being refined manually within these top-level SBs. Based on the behavior described here, this does not seem possible.

The `Target` item in the JTree shows a summary of all items in the `Resources` subtree. Having such a view of all data in a single place is a nice feature. There is however a problem of layout. All the resources are packed very close together, making for a very very dense set of widgets (Figure 7). **S(easy, moderate):** put some empty space between the three different elements. Give the user some space to breathe (visually speaking). With the current layout the user cannot easily differentiate the different logical groups of the UI.

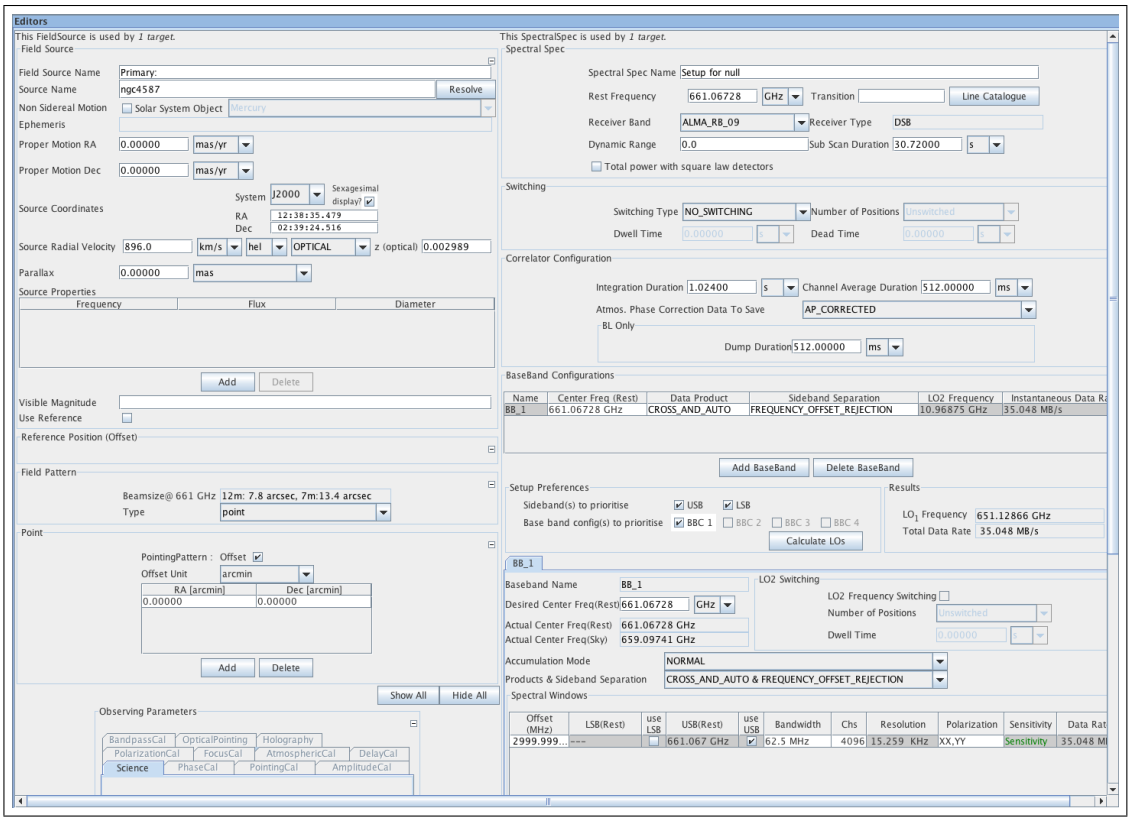


Figure 7: Target and Resources: the very dense layout impedes legibility

3.3 Spectral Tab

Interviews conducted with astronomers at the OSF revealed that this is a very important part of the interface. Astronomers see it as an essential tool to do some exploration, for instance, to identify what spectral lines they could observe in addition to the main one(s) – knowing that there are constraints on what you can observe (receivers, etc.). Most comments are related to the `Visualisation` component of that tab – Figure 8-a.

Generally speaking, this part of the interface has been enhanced since we first saw it in July 2009. Navigation, e.g., pan & zoom, seems to be smoother. There are still some issues though.

One issue that is easy to address is the color scheme used in the plot. **S(easy, moderate):** following Tufte's principles [34], the absorption plot should be rendered with a low-contrasted shade of gray instead of the current highly-saturated purple hue. The absorption plot is an important piece of information, but it does not play an essential role in the interactive specification process. It is more like reference data that should always be visible, like a grid in a CAD program. See Figure 8-b for a simulation of what I suggest (probably better to look at it on-line rather than printed).

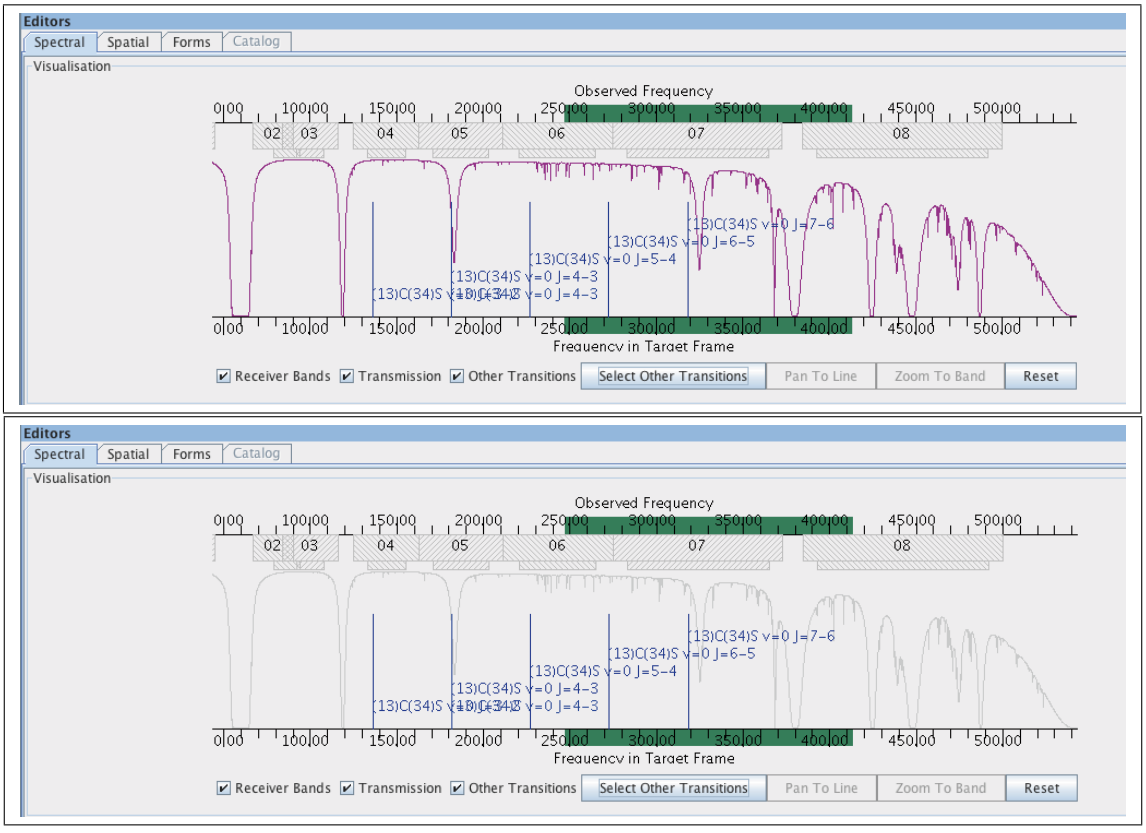


Figure 8: Visualization of Spectral Lines - (a) current version, (b) dimming the absorption/transmission plot so that other components of the plot such as spectral lines better stand out. This might not be very convincing when printed on paper. Please refer to an on-line version.

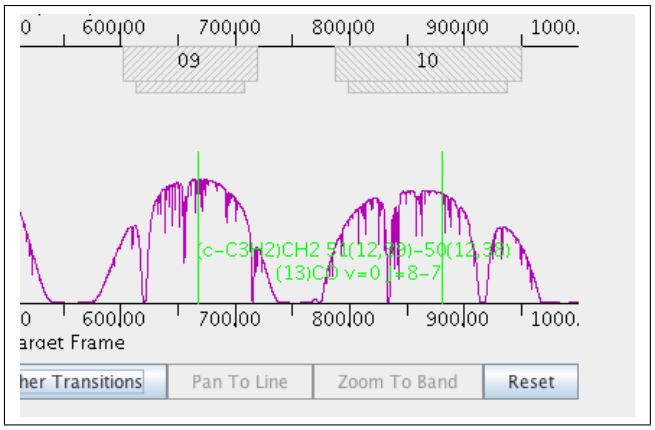


Figure 9: Visualization of Spectral Lines - transitions displayed with a light green color on a gray background are difficult to read

Some of the other colors used in the plot are not legible. Some spectral lines are displayed in light green, and are almost illegible on the light gray background (see Figure 9). Some lines are blue and stand out much better, though it is generally advised not to use tones of blue for thin objects in GUIs. **S(easy, moderate):** pay careful attention to the colors used in the plot. With all the overlap, colors have to be easy to visually differentiate.

Another issue with the plot is that the labels of spectral lines can potentially overlap, making them illegible. I believe this has gotten better than what we saw in July, but there are still issues. When zooming in, spectral lines will get farther away from one another; that will leave more screen real-estate for each label, and eventually, as one zooms-in, there won't be any overlap between labels. But pan & zoom navigation is fairly tedious when what you are interested in are just line labels. Further enhancing pan & zoom interaction as described later would make navigation less tedious, but will not totally solve the problem. **S(moderate, moderate):** make the spectral lines in the plot sensitive to mouse events. When a user clicks on one line, or better, when she just hovers over one line, that line gets highlighted (change its color, make it thicker, or any other visual feedback) along with its label, and the labels of all other lines are temporarily removed from the display. If you'd rather keep all labels visible at all times, then **S(moderate, moderate):** devise a layout algorithm that computes label positions so that there is no overlap at all (e.g., using excentric labeling [8]), even if it means moving the label away from the spectral line, and add low-contrasted diagonal segments that link the label to the actual line on the plot. In this case, visually highlighting the line and label currently hovered by the mouse cursor is still a very good idea.

Last comment about the plot: it is not clear to me (and was not to the people interviewed at the OSF either) what the top and bottom green lines moving in sync with the cursor mean (see Figure 10).

S(easy, moderate): Ignore the parentheses in Filter/Species text area of the `Select spectral lines` window (Figure 11). We were told that astronomers are not used to parentheses in the syntax and will likely type expressions that do not contain parentheses. They should not be forced to input the parentheses. The problem is that if parentheses are omitted, the expression does not match anything (Figure 11-a vs. 11-b).

In both versions of the OT that we saw in July and and in December at the OSF, navigation capabilities in the visualization were quite limited, significantly impeding astronomers' exploratory processes. Things

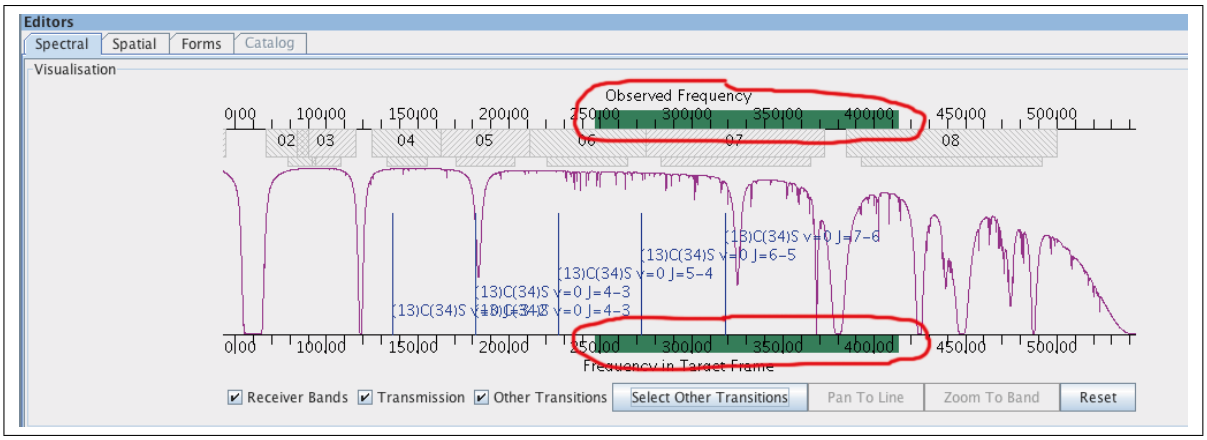


Figure 10: Visualization of Spectral Lines - green bar moving in sync with the cursor

have been greatly improved in v7.0RC. It is now possible to freely pan & zoom in the plot. Zoom level can be changed with the mouse wheel. There are still a few issues that could be addressed to make it even better, as described below.

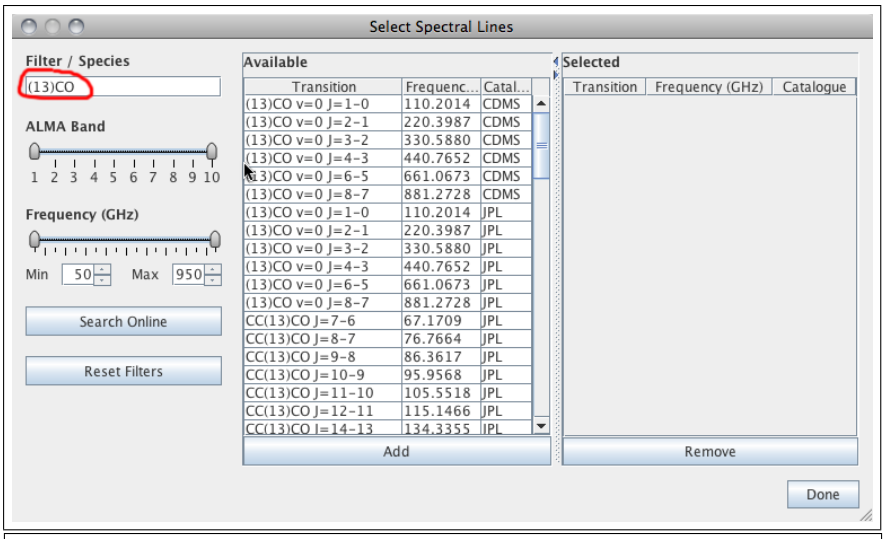
Panning is achieved by dragging the mouse horizontally in the upper and lower regions surrounding the plot itself. It does not work when dragging the mouse in the plot itself. **S(easy, moderate):** allow panning when dragging the mouse in the central region that corresponds to the plot itself.

The mapping between cursor translation and plot panning is not 1:1. Depending on the zoom factor, moving the cursor by N pixels will result in a displacement of the plot that is often significantly less than N pixels. This breaks the impression of direct manipulation [31], but more importantly it can make panning tedious, requiring large mouse/trackpad movements, and thus clutching. **S(easy, moderate):** make panning aware of the current zoom level so as to have a 1:1 mapping between cursor displacement and plot displacement.

There are two main behaviors when zooming in/out: either A) the center of the screen is the focus of expansion, or B) the point under the cursor is the focus of expansion. When zooming in 2D, B) is more powerful but also more difficult to get used to than A). When zooming in 1D, as is the case here (the plot only gets stretched horizontally), B) is still as powerful, and is much easier to master; it actually feels more natural. **S(easy, low):** keep the location under the cursor invariant when zooming in/out, instead of the current plot viewport's center. In other words, the plot should expand/contract around the imaginary vertical bar that corresponds to the current cursor location, instead of the imaginary vertical bar located at the center of the plot's viewport.

S(moderate, high): A more ambitious goal would be to support dynamic queries in the spectral tab [32] to further assist astronomers in their exploration of possible observations. Right now spectral lines get selected in an independent window where they are listed in a table, and then get visualized in the plot when the user is done with that selection process. The two parts of the interface could be made more synchronous. Selections in the `Spectral Line Selector` could automatically appear in the plot (only temporarily if just hovered by the cursor in the table, and permanently if actually put in the selection list). Conversely, selecting a frequency range in the plot, either through a manual selection or by selecting

(a)



(b)

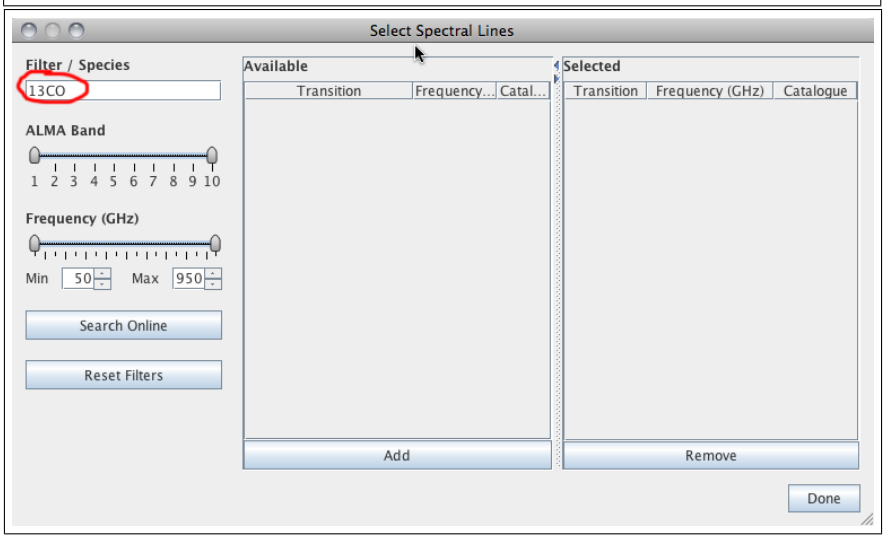


Figure 11: Spectral Lines Selector: (a) matches transitions when putting parentheses, (b) does not if parentheses omitted

one particular receiver band could highlight lines in this frequency range in the table of available lines.

Last comment: in an earlier version of the OT, the `Spectral Line Selector` was triggered by clicking on checkbox `Other Lines`. Clicking on that box would also change the status of this checkbox, which controls the visibility of other lines in the plot. In 7.0RC this part of the interface has been redesigned and does not feature this unconventional behavior any more. I just want to emphasize that this type of behavior with side effects should be avoided.

3.4 Spatial Tab

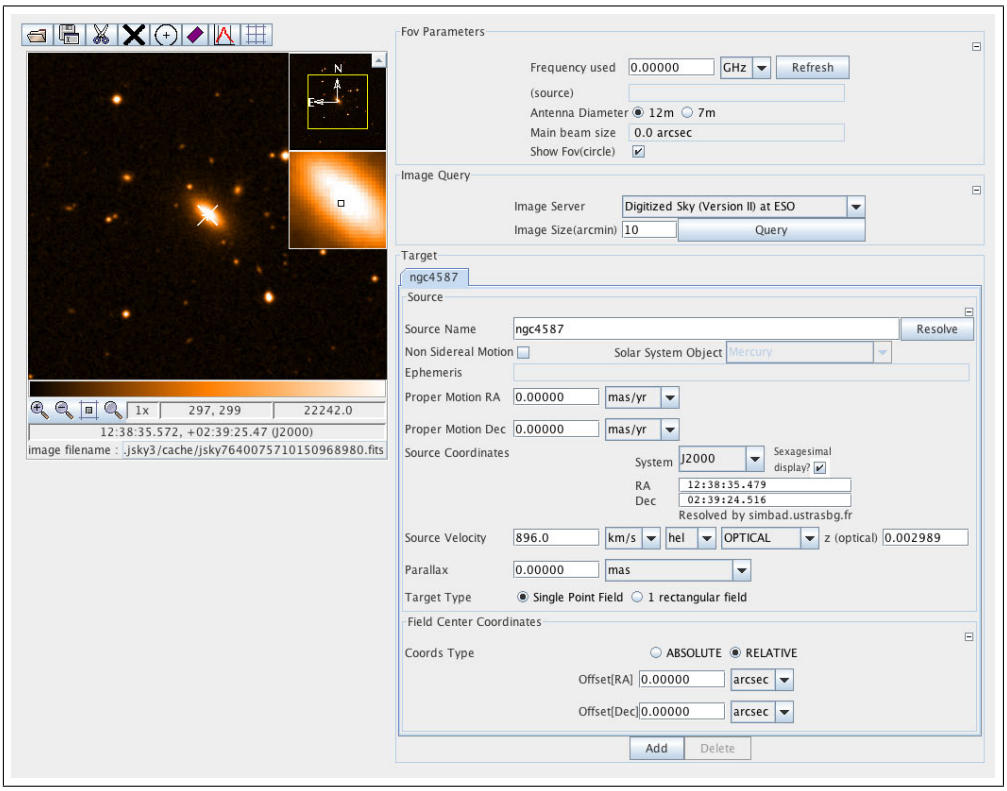


Figure 12: Source Visualization

The `Spatial` tab is used to precisely specify the observation target. This can be done by entering parameters in the `Target` UI component set and by interacting with a visual depiction of the target and its vicinity (retrieving FITS images from a server). Comments about tab management and offscreen content navigation facilities (expand/collapse buttons and scroll bars) have been made earlier in this document. The following comments apply to the FITS visualization panel.

This panel (Figure 12) is very reminiscent of ds9's interface (Figure 13). This is both an advantage and a disadvantage. It is an advantage because many astronomers use ds9. Most of them will thus be familiar with multi-scale interfaces based on a main viewport plus a magnifier plus a panner. This is a disadvantage because this type of UI design is old and does not scale. For instance, having the magnifier as a static inset means that it is not smoothly integrated with the visualization in the main viewport, and that it is potentially far from the cursor (which controls the content of the magnifier, as the latter shows a magnified version of region surrounding the cursor in the main viewport). This causes problems of divided attention [6], that can be solved by using state-of-the-art focus+context visualization techniques such as Sigma lenses [25, 26]. Similarly, the panner could be enhanced, so that it behaves more like current overview+viewfinder widgets [21] (see Google Maps for an example). Of course, if FITS images to be visualized in the spatial tab are small and do not really require the use of the panner and magnifier, then the current UI layout and features will do. But if it is anticipated that relatively large FITS images will be loaded, then **S(moderate to**

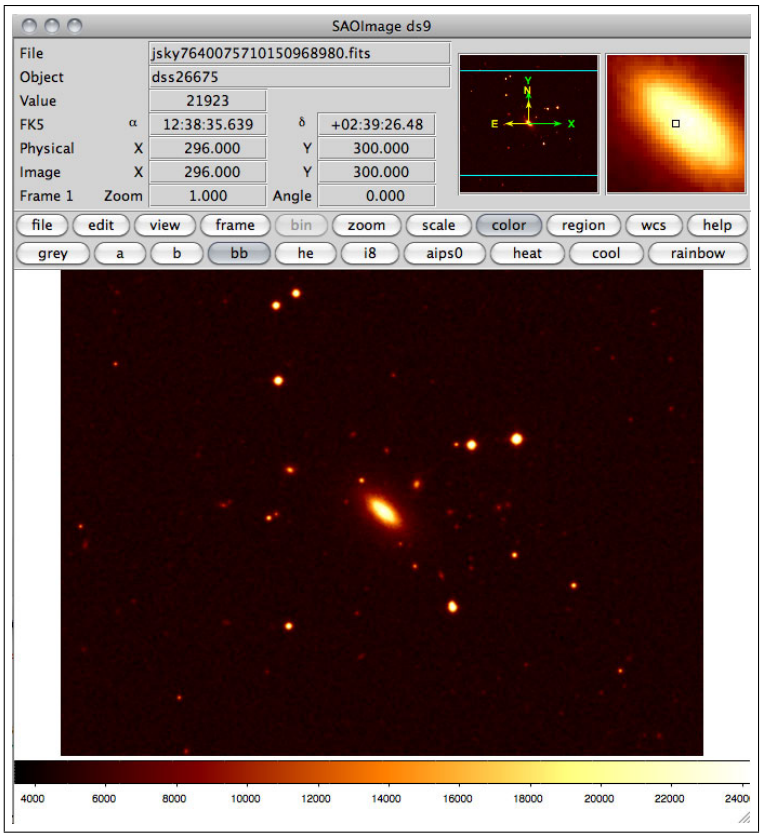


Figure 13: Visualizing a FITS image in ds9, with a similar interface

hard, moderate): use state-of-the-art multi-scale interaction techniques that will make it easier for users to navigate in the image. Difficulty is rated as moderate if you use third-party Java UI libraries that provide off-the-shelf visualization components that integrate the above-mentioned techniques, e.g., [24, 35], and is rated as hard if you do it from scratch with Java2D+Swing.

S(easy, low): map mouse wheel events to zoom in/out in the main viewport, in addition to the magnifying glass icons in the bottom-left corner.

3.5 Miscellaneous

S(easy, high): It takes time to edit a proposal and the task is a complex one. Users do not want to have to do it all over again in case a problem occurs. Even though losing data/documents is a source of great frustration, people tend not to save very often. Auto-save/auto-recovery mechanism, as found in MS-Office like programs (Word, Excel, etc.), are very useful in this respect. There does not seem to be such a feature in the OT. It would make sense to add it so that users can restore the proposal to a state close to what it was prior to the occurrence of problems such as application/system crash, power failure, etc. Since open/save/import/export features have already been implemented, this should be relatively straightforward.

S(moderate to hard, moderate): there is no undo/redo mechanism. The interface mostly follows the *form fill-in* type of interaction style [33], and in this context undo/redo is not a fundamental feature, because of the impact of changes often localized to the field being edited. However, it is still easier for the user to fix an error by reverting back to the previous state using an *undo* button rather than having to remember what was the previous value and input it again (which is much more cognitively demanding). Difficulty is rated moderate to hard because from a software engineering perspective, if undo/redo has not been planned early in the design process, it can be difficult and time consuming to add support for it (this also depends on whether a single-step undo or a history of the N last actions is offered).

S(easy, moderate): on multiple monitor (multimon) configurations, popups and new windows can potentially appear on another monitor than the one displaying the OT. This is a small but very annoying problem [12, 17] that is relatively easy to fix: it should be possible with Swing to get the ID of the screen device hosting the OT's main JFrame, and then relocate popups when they appear, based on that information.

S(easy, low): toolbar buttons should be grouped logically. Right now they are all juxtaposed (Figure 14-a). Insert some empty space between groups of buttons that are part of the same logical groups, as done, e.g., in Figure 14-b.

(a) (b)

Figure 14: Group items in the toolbar according to their semantics, as in (b)

S(easy, low): why is the `Abstract` field in `Proposal information` only editable through a popup window (Figure 15)? All other fields can be edited directly, text can be copy/pasted in them, except for this one. The user has to pop up a window, type her text, and then click ok. This is cumbersome. If this was done to enforce the 300-word limitation, you can check that dynamically as the user types in the main text area, and give feedback in there about potential overflows.

S(easy, low): avoid using Italic fonts in GUIs. The DPI (resolution) of most screen panels is too low to make italic fonts easy to read (Figure 15).

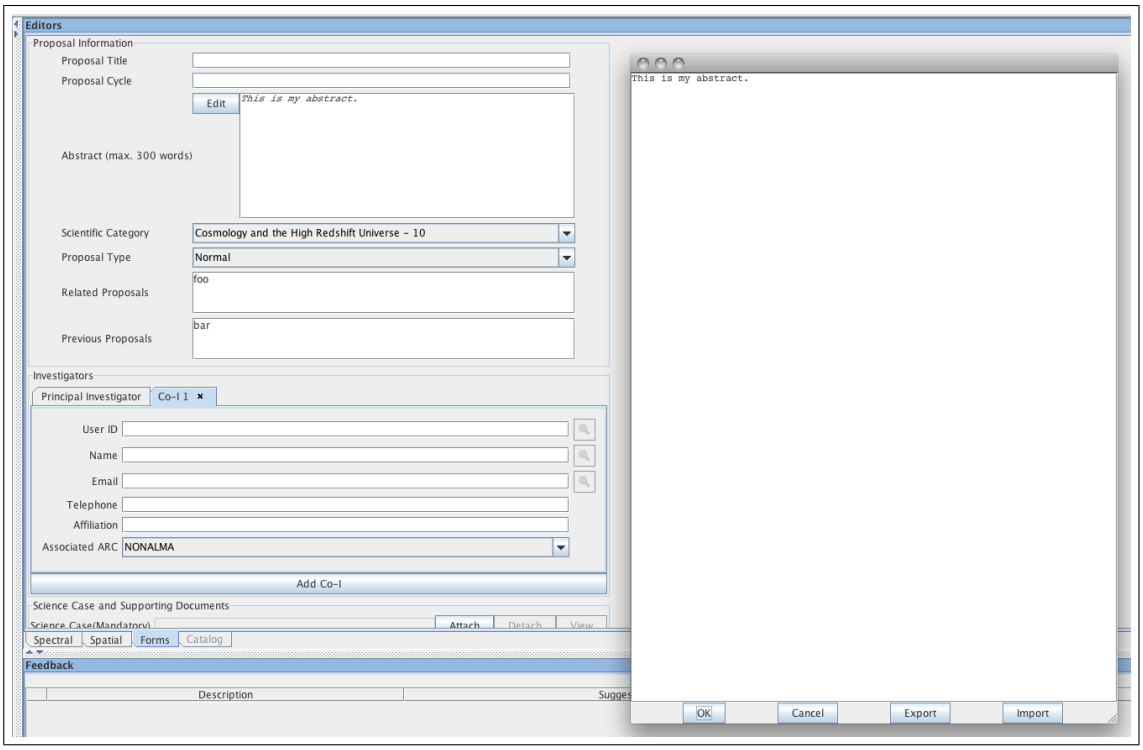


Figure 15: Metadata associated with the proposal

4 Executive Subsystem

Contact Author: emmanuel.pietriga@inria.fr

Syntax used for suggestions: **S(difficulty, impact):** ...

4.1 Overall Organization of the UI

The Operator Monitoring and Control (OMC) application [10] is intended for operators and astronomers on duty. It lets them startup the system, create arrays, manage them, monitor and control devices, the execution of scheduling blocks, etc. Many parts of ALMA can be accessed through the OMC via plugins, including logs, alarms, antenna device status, software components, etc.

From a usability perspective, the OMC is very different from the OT, in the sense that it will be used by a small number of expert users that will be exposed to it for long periods of time, daily. They can thus be expected to behave as power users [33], that take the time to learn how to operate the software and use all of its features, even if the learning curve is steep. This does not mean that the UI design does not matter. Power users have the capability to adapt to complex interfaces, if only because their job requires it, but bad UI design can have strong negative consequences. This is not a question of user comfort: a difficult-to-use, over-crowded UI will be the source of more errors; it also means that users are more likely to fail to react quickly enough to a problem, simply because they are cognitively overwhelmed with data and because their interaction with the software is impeded by tedious UI navigation tasks. For critical and complex systems such as ALMA, careful UI design is very important, even if the users are assumed to be power users.

I start this review with the overall organization of, and navigation in, the OMC interface. Subsequent sections each deal with a specific interface component or plug-in (including those that can also run stand-alone such as JLog).

The interface consists of:

- A thin menu/status bar at the top that gives concise information about the overall system's status.
- A tabbed pane with one pane for the overall ALMA system, and one tab per array created. Each of these panes can contain an arbitrary number of internal panels that correspond to the various OMC plugins. The layout of these panels can be reconfigured with conventional window-management techniques provided by the JIDE docking framework.

The docking framework lets users manage UI panels through the metaphor of windows arranged into space-filling layouts through drag & drop. It is relatively powerful, featuring capabilities such as insertion of panels into existing components as new tabs. Another advantage is that most users will be familiar with this style of interaction, fairly similar to how users manage windows on the desktop. However, this conventional, WIMP-based solution [7] does not scale to the large number of plugins and antennas that have to be dealt with in the OMC. The docking framework gives users a lot of flexibility, but this flexibility has a non-negligible cost: because of the number of panels and antennas, users will spend a lot of time managing windows, because only a very limited subset of all information of interest when operating the

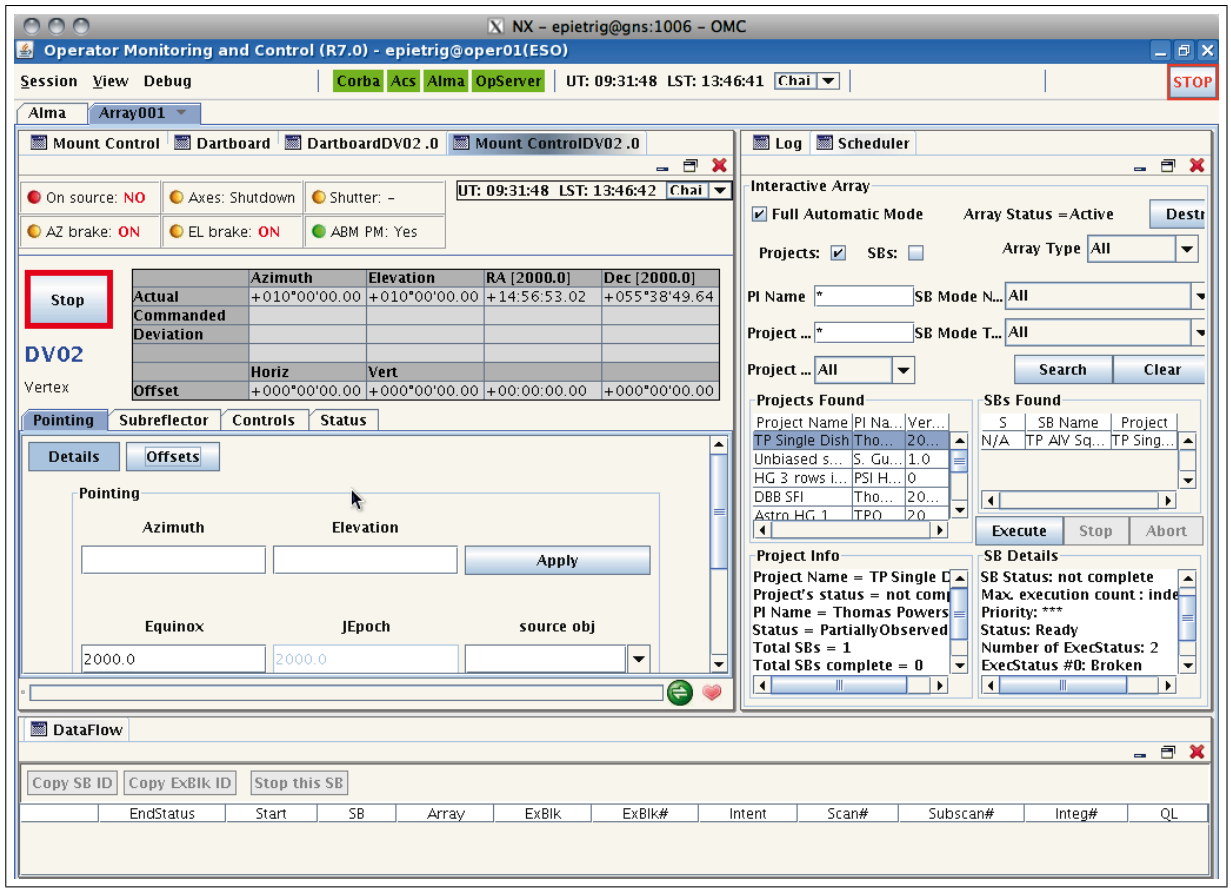


Figure 16: OMC interface with one array, showing the Scheduler plugin, the Mount Control plugin for one antenna, and the Dataflow plugin

array can fit on screen at a given time, even when using three or four displays per workstation. This will seriously impede navigation in the interface, and consequently operator/astronomer efficiency. Issues with the docking framework were also raised by a group of early users in 2007 [30] and corroborate our own findings.

I did not have the opportunity to check whether some of these legitimate concerns were addressed or not, such as (A) *"Multiple instances of plugins are not currently possible"* or (B) *"You cannot apply a different layout when the software is already operational"*. (A) is clearly a very significant issue. Reading more recent documents I believe it has been addressed, but in case it has not, it definitely has to be: (A) makes navigation extremely cumbersome and prevents making comparisons between and/or contrasting values for similar components. I believe that (B) has not yet been addressed as I recall operators still complaining about it during interviews at the OSF. This is clearly an issue, as different users will want to customize the interface according to their preferences based on their mental model of the system.

The chessboard is used as an entry point for many plugins to specify which antenna the operator wants

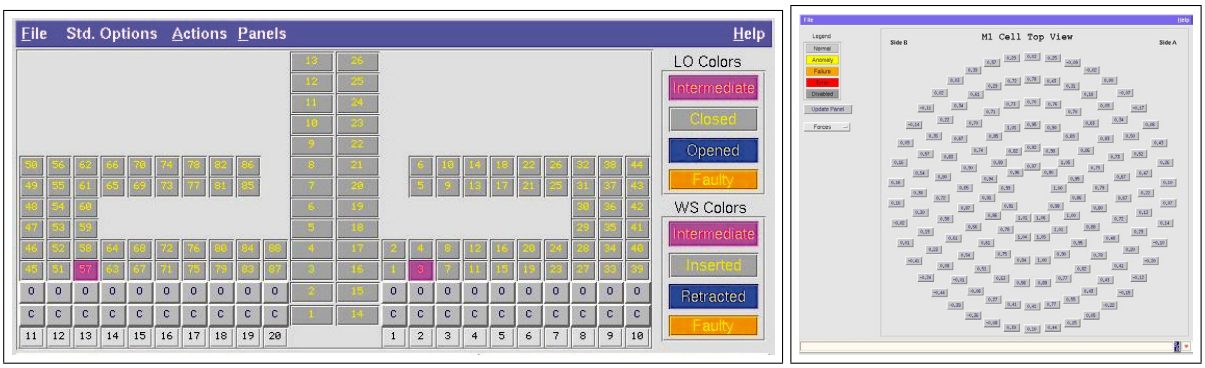


Figure 17: Basic examples of component layouts that help users build a stronger mental map of the system (Paranal/VLT).

to get detailed information about. It is a very compact widget, that does a good job overall[1]. However, the OMC's interface lacks a representation of arrays – and of the entire ALMA system – that is closer to their actual physical/geographical layout. This type of visualization should not be neglected. It greatly helps users build a clear mental model/mental map of the system, which then allows them to more efficiently navigate in the interface, leveraging low-level cognitive abilities such as spatial memory and spatial orientation. Such capabilities are left totally unexploited by the current UI based on a WIMP docking framework. In the case of ALMA, the advantage of such visual representations (Figure 17 gives basic examples from the VLT) goes beyond this; interviews with operators and astronomers revealed that information about baselines is essential and some of it has to be interpreted relative to the baseline's length or to the meteorological conditions, which can vary from one section of the site to another (from which I infer that the geographical location of antennas is not an irrelevant piece of information). Antennas form the nodes of a network whose edges are the baselines. This network is an important abstraction of the system, and representing it through an interactive visualization component would both make navigation in the system more efficient and further help users build a strong mental map of the system, simply because it explicitly represents information that is currently only conveyed through text, making mental tasks such as comparisons more cognitively demanding.

S(hard, major): All these issues represent a major UI challenge that cannot be addressed by incremental enhancements. The current layout is too abstract and volatile. It requires re-considering the overall organization of the UI, in favor of more advanced information presentation and navigation paradigms such as, e.g., multi-scale user interfaces [1, 9, 16, 23] and multiple coordinated views [22], that will enable a more efficient use of available screen real-estate and will let users navigate more efficiently in the information space, including support for custom groupings of components in "compound objects" as described in [30].

Re-considering the overall organization of the UI does not require completely reimplementing the OMC. Many of the plugin panels can be kept almost "as is", plus the implementation of suggestions made in the following sections (most panels can run as plugins or as standalone applications, meaning that from a Swing UI implementation perspective they should be easy to adapt to any Java2D-based graphics framework). What requires extensive work is the general information presentation and navigation metaphor

[1]The current version is buggy however: popup chessboard panels are initialized to a very small size where it is impossible to read any antenna name, requiring users to resize them every time they pop up.

based on the WIMP docking framework (JIDE). During my stay at the OSF I implemented a quick and dirty prototype of multi-scale interface based on the geographical layout of antennas, integrating device panels in the zoomable canvas and showing the network formed by antennas and baselines (Figure 18). The purpose of this small prototype was just to show the audience that other interface metaphors – that better leverage human abilities, and can show more structured information while providing more efficient means of navigation – are available. Obviously a solution based on the layout of antennas based on the exact geographical position of antenna pads is not optimal, because the pads are scattered too thinly over a large area. But I believe that a mix of multi-scale interface with embedded conventional WIMP components (plugin panels) and multiple coordinated views would represent a major step forward and allow the OMC to scale to 60+ antennas. Designing and implementing such a solution requires significant work: participatory design workshops with operators and astronomers, design of appropriate visualizations and associated interactions, selecting, learning and using a multi-scale structured graphics + infovis toolkit, and integration of plugins, including support for communication between some of them (see later comments about the lack of synchronization between some plugins).

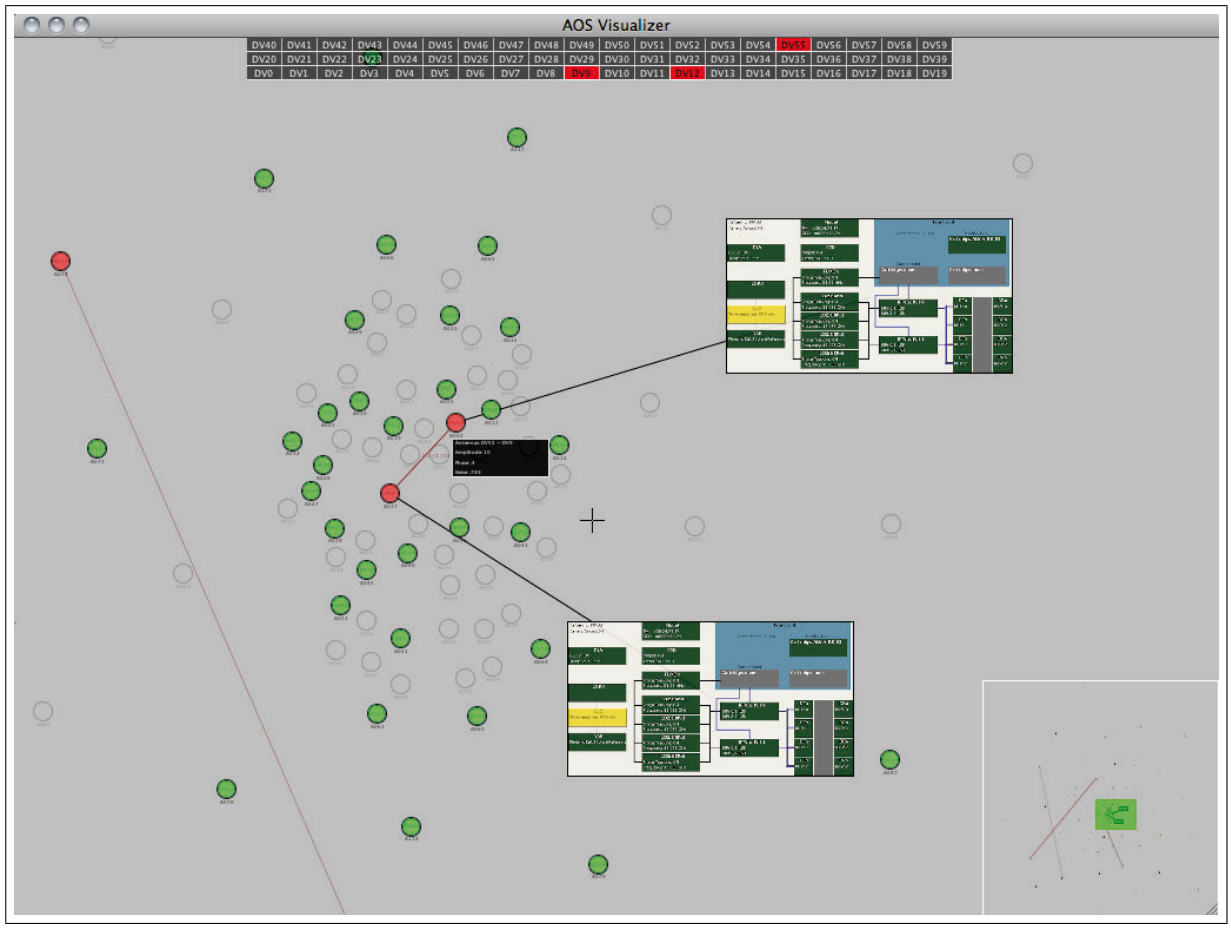


Figure 18: Multi-scale interface prototype with embedded device block diagram and baseline info

S(moderate, high): A *UI component finder* would also enable users to more efficiently navigate in the interface: just by typing keywords and component IDs in a text field, the appropriate UI component would be triggered. For instance, to bring up the `DTS Digitizer Clock` for `DV01`, the user would just have to:

1. trigger the UI component finder with a keyboard shortcut
2. type `dgck dv01` or any variation (sensitive neither to case nor to token ordering)
3. hit enter

instead of having to:

1. reach (physically) for the mouse
2. go to the `View` menu
3. click on it
4. visually scan the list for `DGCK`
5. move the cursor down to that item
6. click the left mouse button
7. move the cursor to cell `DV01` in the chessboard that popped up
8. double click on it

Of course, the *UI component finder* would require the user to memorize some terms. But operators and astronomers, as expert users, will already be familiar with these terms as they are continually exposed to them; thus this should not be a problem.

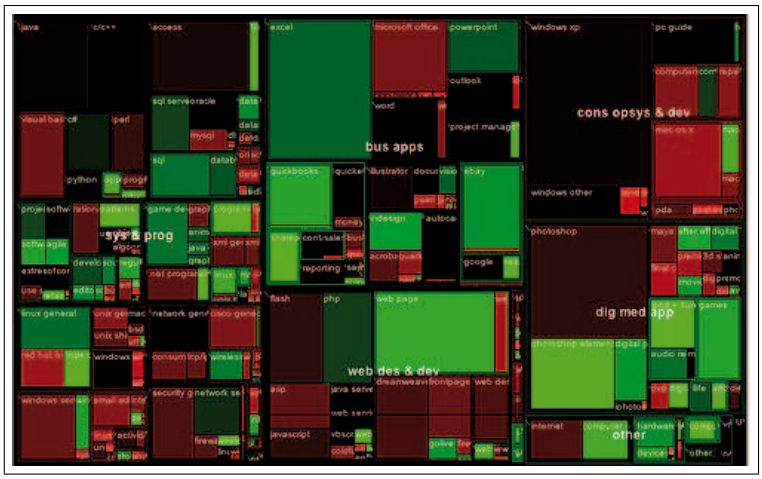


Figure 19: Hierachical structure represented as a treemap: this visualization provides a compact yet legible view of the tree's components.

Finally, **S(moderate, high):** Treemaps [2] would be efficient replacement for several hierarchical structures currently represented with JTrees, such as in the `ACS Overview panel`. Treemaps (see an example in Figure 19) provide a compact yet legible view of the tree's components without requiring much interaction, ideal for monitoring the status of nodes located at different levels of a hierarchy. Combined with the smooth zooming capabilities mentioned above, treemaps can be a very efficient means of navigation in hierarchies such as those found in the OMC.

4.2 ALMA Tab and Creation of Arrays

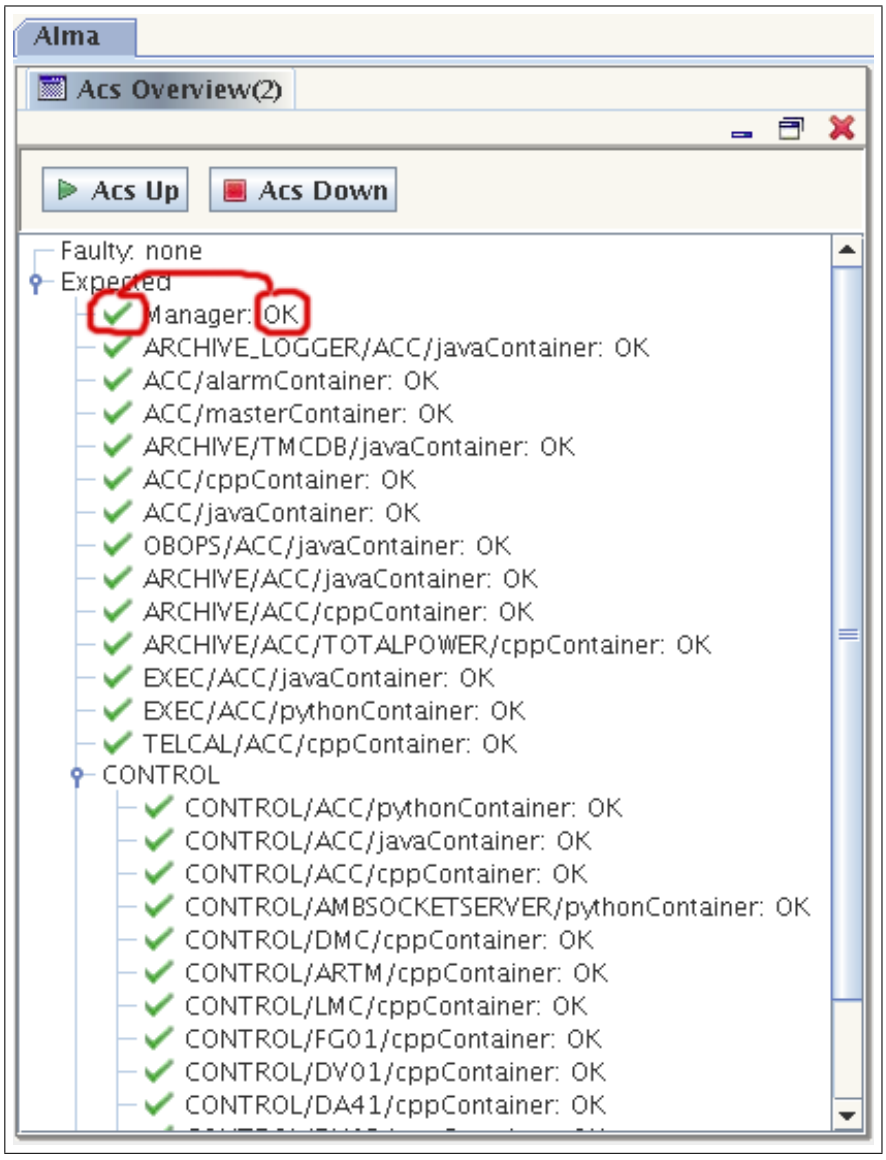


Figure 20: JTree used for containers in ACS Overview panel

In the ACS Overview panel, it is not clear how redundant the green ticks and the `OK` strings are (Figure 20). **S(easy, low):** If there is a single icon corresponding to each possible status string, the status string could be discarded, making each item shorter and decreasing the overall complexity (of course, if the mapping between icons and status strings is not 1:1, this suggestion does not make sense). **S(easy, low):** if it is expected that the containers will be started/stopped often, provide start/stop icons on each line (as for subsystems – Figure 21) to perform these actions instead of doing this through a popup menu, so as to minimize the number of steps required to perform the action.

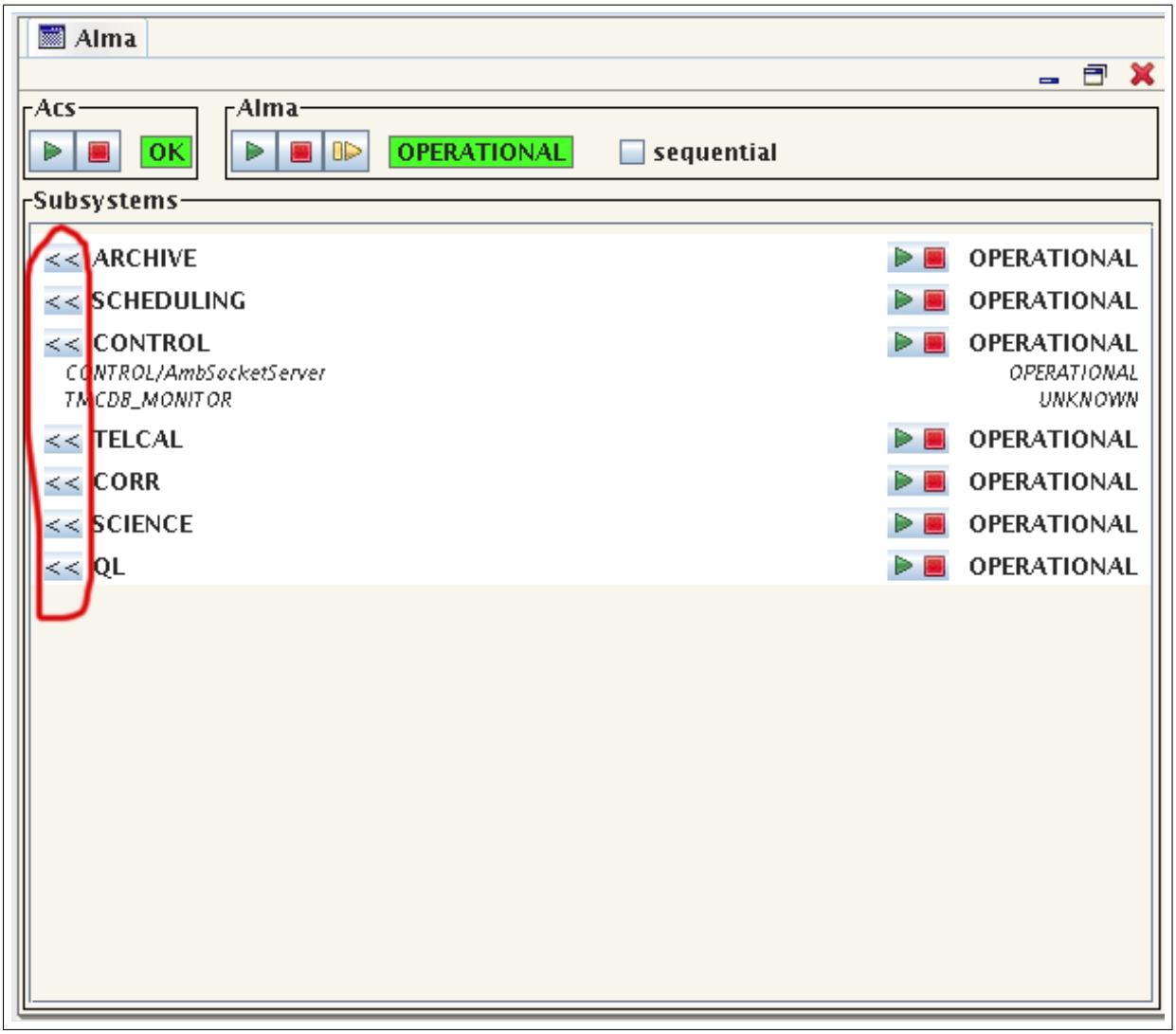


Figure 21: Unconventional contract/expand icons

The ALMA Subsystems panels use unconventional icons to show/hide detailed information about each subsystem (Figure 21). **S(easy, low):** Unless this particular couple of icons bears explicit semantics that operators and astronomers have seen before (in other telescopes' operations UI), it would be better to replace it with more conventional icons such as the ones used to symbolize contracted/expanded branches of a JTree (or file/folder tree in Windows Explorer-like expanding menus).

Device Explorer panel: according to [10], Section 6.4.2: *Each node has a tool-tip text which provides the explicit state of the node and some additional information*. As detailed in Section 4.7 about QuickLook, tooltips pose problems of persistency and delay, and in the case of this panel do not allow users to get an overview of the state of nodes (the user has to hover one particular element, and can thus only see one at a time). **S(easy, moderate):** Provide essential status information in the node's label.

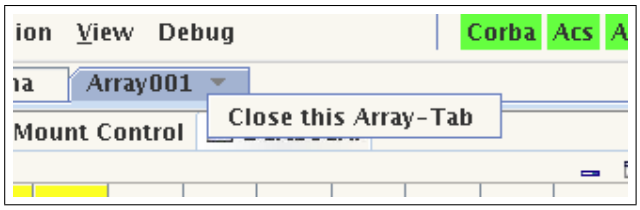


Figure 22: Array tab management

S(easy, low): Array tabs all contain a contextual menu triggered by clicking on the arrow next to their label (Figure 22). This menu only features one action, that corresponds to closing the tab. If this action is the only one expected to appear in this menu, replace the arrow by a cross symbolizing tab closing similar to the tab management interface of many tab-based interfaces such as Web browsers (see Figure 5) for an example). This is to minimize the number of steps required to perform the action. A similar comment can be made about the table in the Create Array/Existing array panel (destroying arrays performed by popping up a contextual menu for each row).

4.3 JLog

This tool provides a unified interface to monitor logs coming from many sources. The application is able to handle large numbers of log entries, thanks to an advanced, multi-level caching system [27].

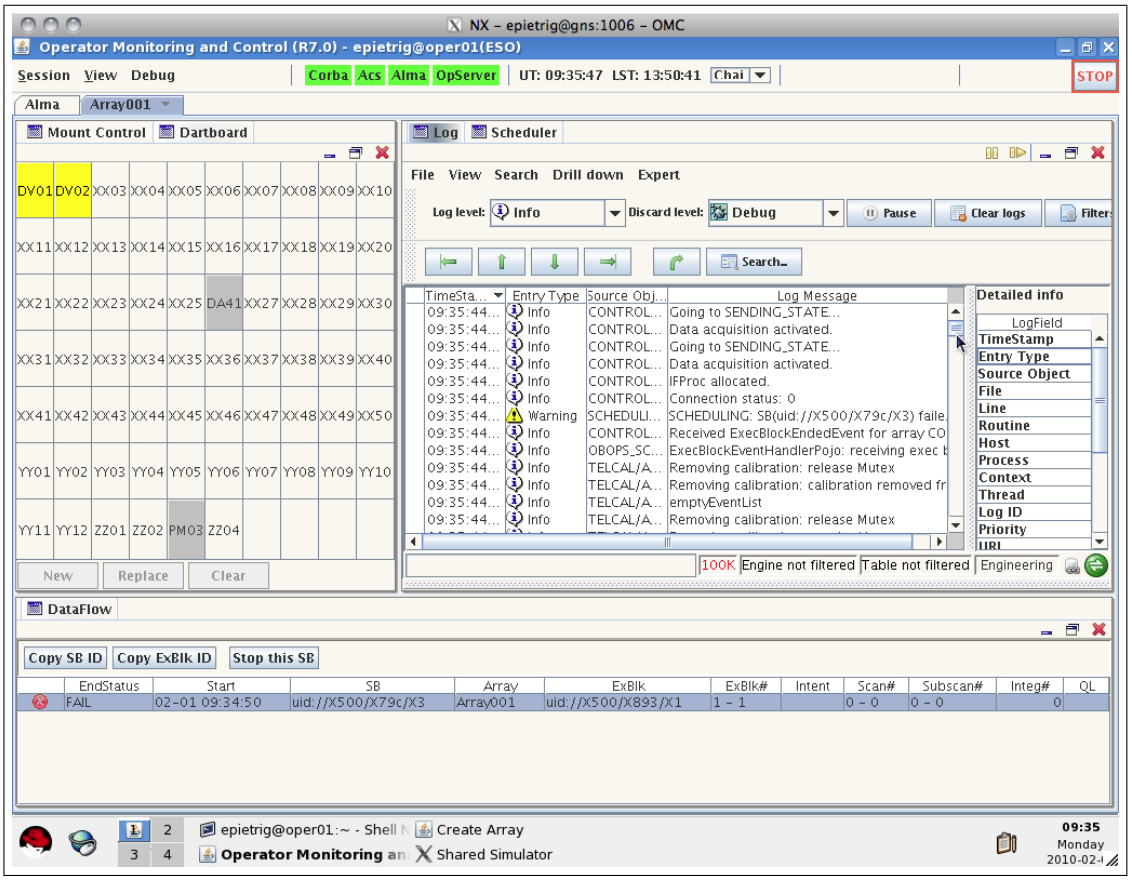


Figure 23: JLog running as a plugin

S(easy, moderate): the toolbar is currently split on two lines. When JLog runs standalone, this is ok. But when run as a plugin, this consumes a little too much screen real-estate for the added value. Reducing the width of buttons, it might be possible to fit all of them on a single row, maybe removing some of them that could be less often accessed than others (`Clear logs` ? `Filters` ?). There is also a problem of accessing buttons on the far right if the plugin's panel is not wide enough (see Figure 23).

S(moderate, moderate): Change `Log level` and `Discard level` widgets from drop down lists to sliders for more efficient switching between levels. Changing the UI widgets is straightforward. However, if moving to sliders, the UI has to be very responsive to changes. In other words, the log table has to update quickly to the changes. The caching system might be leveraged to achieve a good level of performance for this more dynamic type of query. Ideally, a more interactive search paradigm, such as dynamic queries [32], or even basic suggestions as is now commonly found in Web query interfaces, would make the search/filtering more efficient. The log table can be very large. Dynamic queries would clearly improve

search/filtering efficiency, but might also be challenging to implement. Note: in both cases, if the system cannot keep up with user input (if it cannot refresh the table fast enough), there is no point in doing this. It would actually be harmful as users will not expect any lag in dynamic query interfaces. **More on search: S(easy, moderate):** searching for a particular pattern only takes the user to the next matching entry. There is no highlighting of the entire set of matches, making it impossible for the user to get an overview of the result set, or quickly scan through it. If the above-described dynamic query style interface cannot be implemented, at least highlight all matching entries in the table when a search is performed.

Drilldown: when a user drills down, i.e., when he asks for detailed logs for a given time period, the rows that were originally selected by the user to specify the time period get unselected, with additional rows being inserted within that range. It makes it very difficult to relate the two states. **S(easy, moderate):** original rows should remain selected or just highlighted by changing their background color, and additional rows that got inserted should be highlighted too, possibly with a different color to make it easy for users to differentiate both types of events (original events, newly inserted ones from the detailed log).

S(easy, low): Table columns can be reordered with a right mouse drag on the column header. This is usually done with a left mouse drag (without colliding with reordering actions).

Filters: [27] Section 3.5 mentions that some combinations of filters can result in empty log tables. **S(easy to moderate, low to moderate):** If this can be checked automatically by the system, it would be good to provide feedback to raise this concern to the user while she is creating her filters.

Regular expressions: S(easy, moderate): regular expressions are fairly easy to learn, but most people are not familiar with the syntax and sometimes confound it with POSIX file system wildcard syntax. Provide the basic but meaningful examples found in [27] directly in the interface through a small contextual help widget, allowing users to copy/paste/edit them.

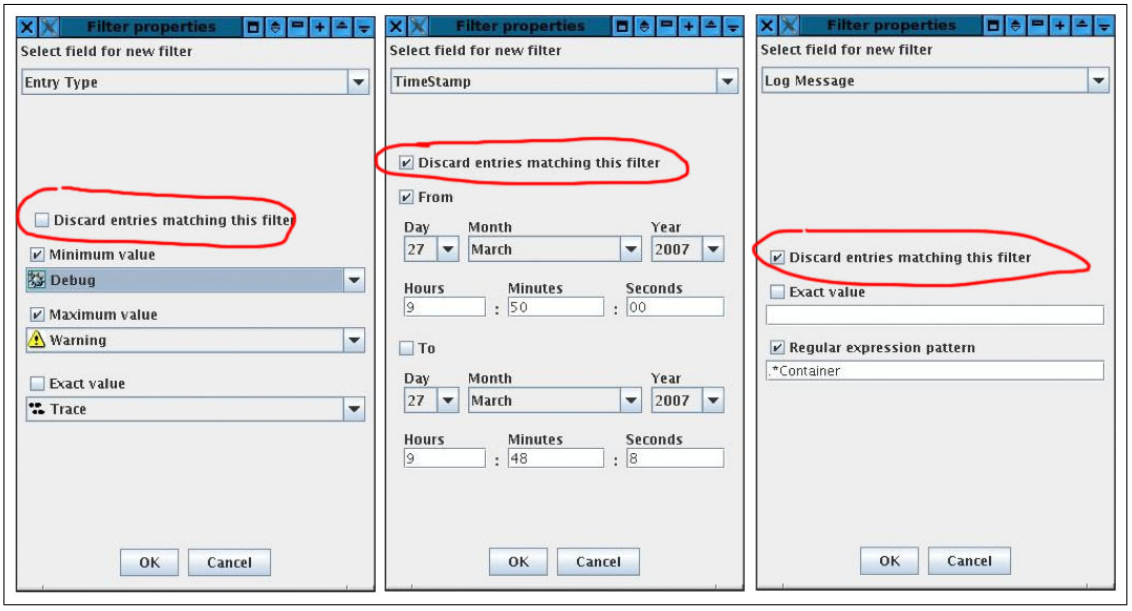


Figure 24: Wrong use of a checkbox for mutually exclusive choice

Filter properties: According to [27] Section 3.5, a checkbox `Discard entries matching this filter` is used to indicate whether the filter should be interpreted as a boolean True or False in the overall AND expression (Figure 24). The use of a single checkbox widget makes its semantics ambiguous. This is a simple case of mutually exclusive choice, and nothing is optional about it (a false impression conveyed by the use of a checkbox). **S(easy, moderate):** Replace the checkbox by two mutually exclusive radio buttons: `Keep entries matching this filter` and `Discard entries matching this filter` or variations on these wordings.

Keyboard shortcuts: as mentioned earlier the OMC is a tool for power users, and keyboard shortcuts will be very beneficial to users[2]. **S(easy, moderate):** Make sure that:

- most often used commands have key bindings;
- input focus sequence between widgets follows a sensible order (Swing `FocusManager`);
- navigation in the table (also available with the toolbar on the 2nd row) can also be performed with key bindings such as arrow keys and page up/down for reaching the top/bottom of the table. Home can be used to reach the selected row when it got out of the scrollpane's viewport.

The **main issue** with JLog, however, is the lack of synchronization with the Alarm Panel (when running as an OMC plugin). Both plugins seem to be totally unaware of each other, requiring users to manually relate entries between them. This seems to be due in part to the fact that it was decided from an architectural point of view that plugins should be totally independent from each other. While this might make sense from a software engineering perspective, it does not from the perspective of the user. I would argue here that the users' needs take precedence over conserving a pure and elegant architecture (as long as it does not negatively impact robustness of course). **S(moderate to hard, high):** provide mechanisms that allow the user to correlate log entries with alarm events, even if only based on time stamps (making the assumption that both systems share the same time reference). What actions are relevant and how this should be done from a UI design perspective requires further discussion with users, but this was clearly identified as a useful feature during interviews. A basic example would be to highlight log entries in a time period focused on (centered around) the timestamp of the selected alarm. More relevant selections can probably be devised.

[2]This is actually a general comment that applies beyond JLog, to all OMC components and to the framework itself.

4.4 Alarm Panel

The Alarm Panel [4] displays events related to malfunctioning software and hardware components.

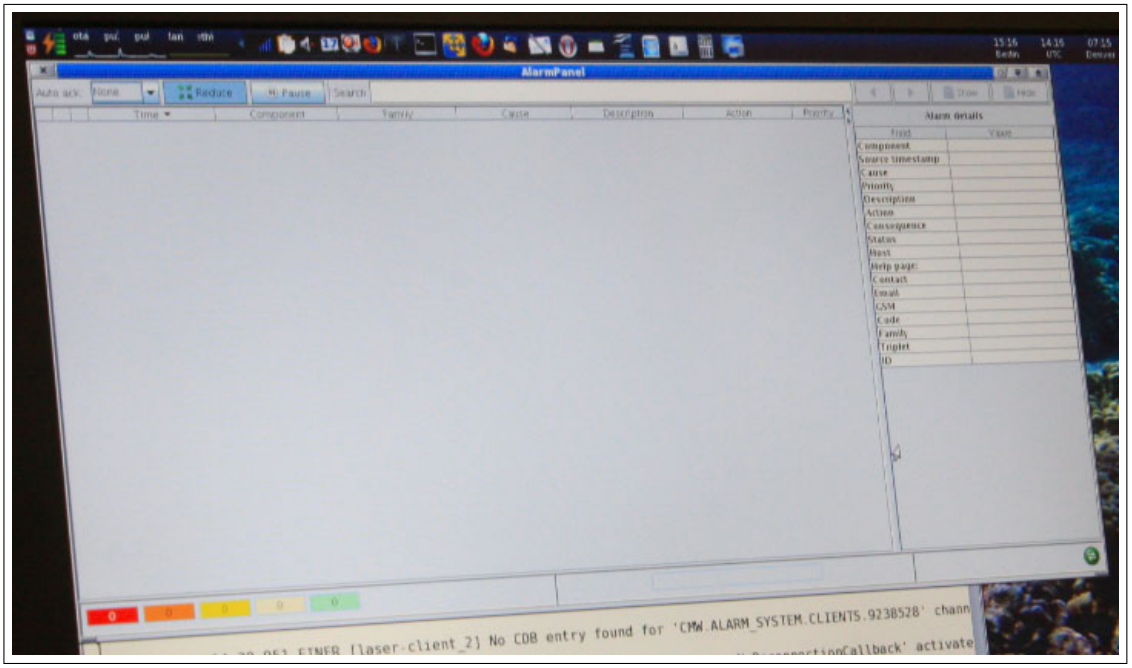


Figure 25: Alarm Panel running standalone

The auto-acknowledge feature is nice, but should be used with caution.

Search functions and dynamic query like interfaces are of lesser importance for the Alarm Panel than for JLog because the number of entries is supposed to be much lower, and thus much more manageable.

Keyboard shortcuts: as mentioned earlier for JLog, the OMC is a tool for power users, and keyboard shortcuts will be very beneficial to users. **S(easy, moderate):** Make sure that:

- most often used commands have key bindings;
- input focus sequence between widgets follows a sensible order (Swing `FocusManager`);
- navigation in the table can also be performed with key bindings such as arrow keys and page up/down for reaching the top/bottom of the table.

S(easy, low): make sure that the ordering of timed events in AlarmPanel and the default ordering of timed events in JLog are consistent.

Again, here the **main issue** is the lack of synchronization with JLog (when running as an OMC plugin). **S(moderate to hard, high):** See comment about this at the end of Section 4.3.

S(hard, high): Coupling of alarm entries with the associated components in a redesigned multi-scale interface as described in Section 4.1 would also be beneficial, better helping the user to situate the problem in its context (some sort of *take me where the problem is and give me the tools to address it* feature).

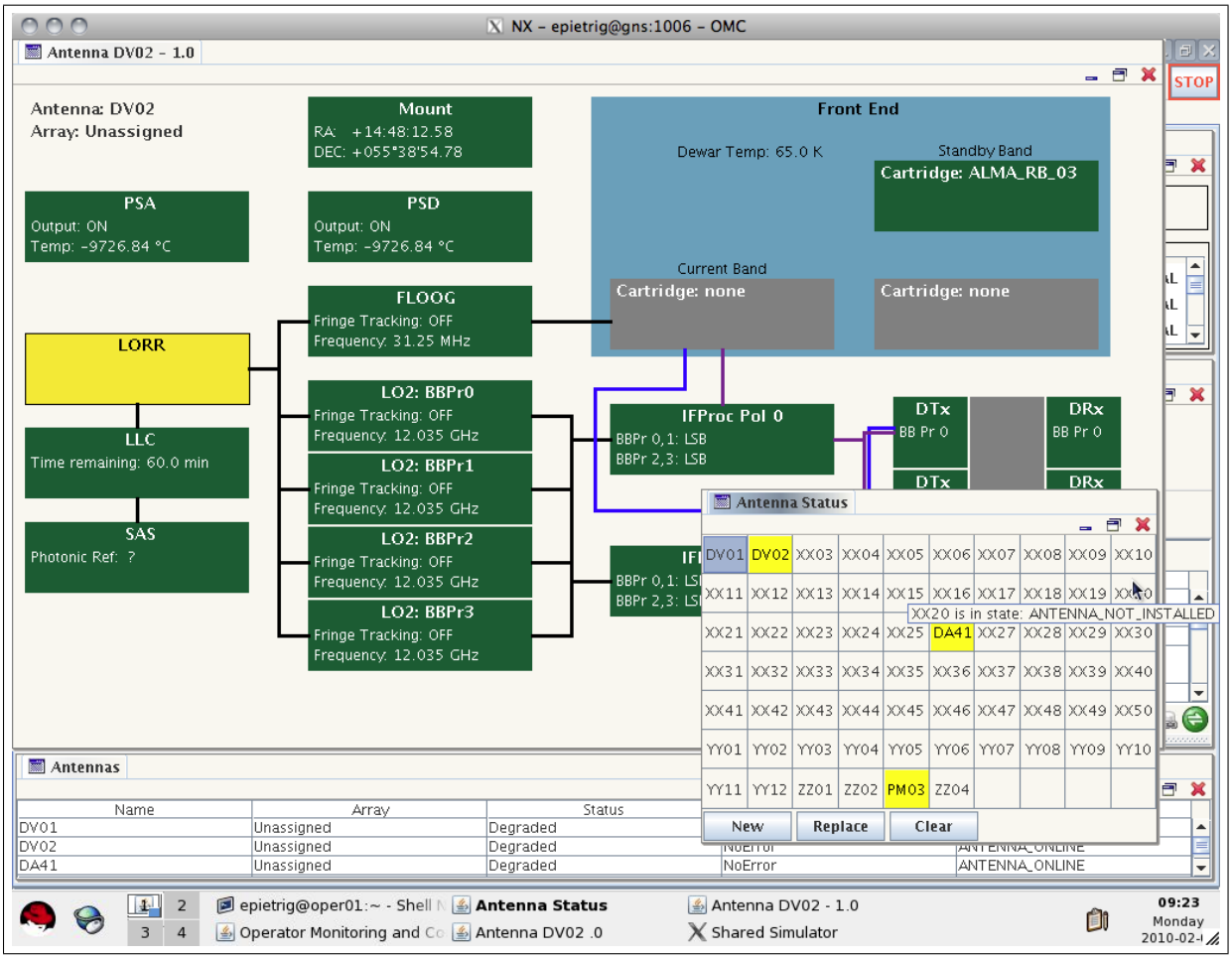


Figure 26: Block diagram of the major devices in an antenna

4.5 Control Devices

A block diagram is used to show the most important devices in an antenna with a summary of their status and an explicit representation of their relationships (Figure 26). This is very appropriate. Similar diagrams are used, e.g., at the VLT for several subsystems (Figure 27). **S(moderate, high):** schematic representations of devices could be used in several device components. Again, such visual representations leverage human spatial memory, reduce visual search time, provide richer depictions of systems thanks to the multidimensional arrangement of information that also enables the explicit depiction of relationships between components (something that cannot be done with text alone), and contribute to reducing the user's cognitive load overall. Figure 28 illustrates this with additional examples from the VLT interface for different hardware components.

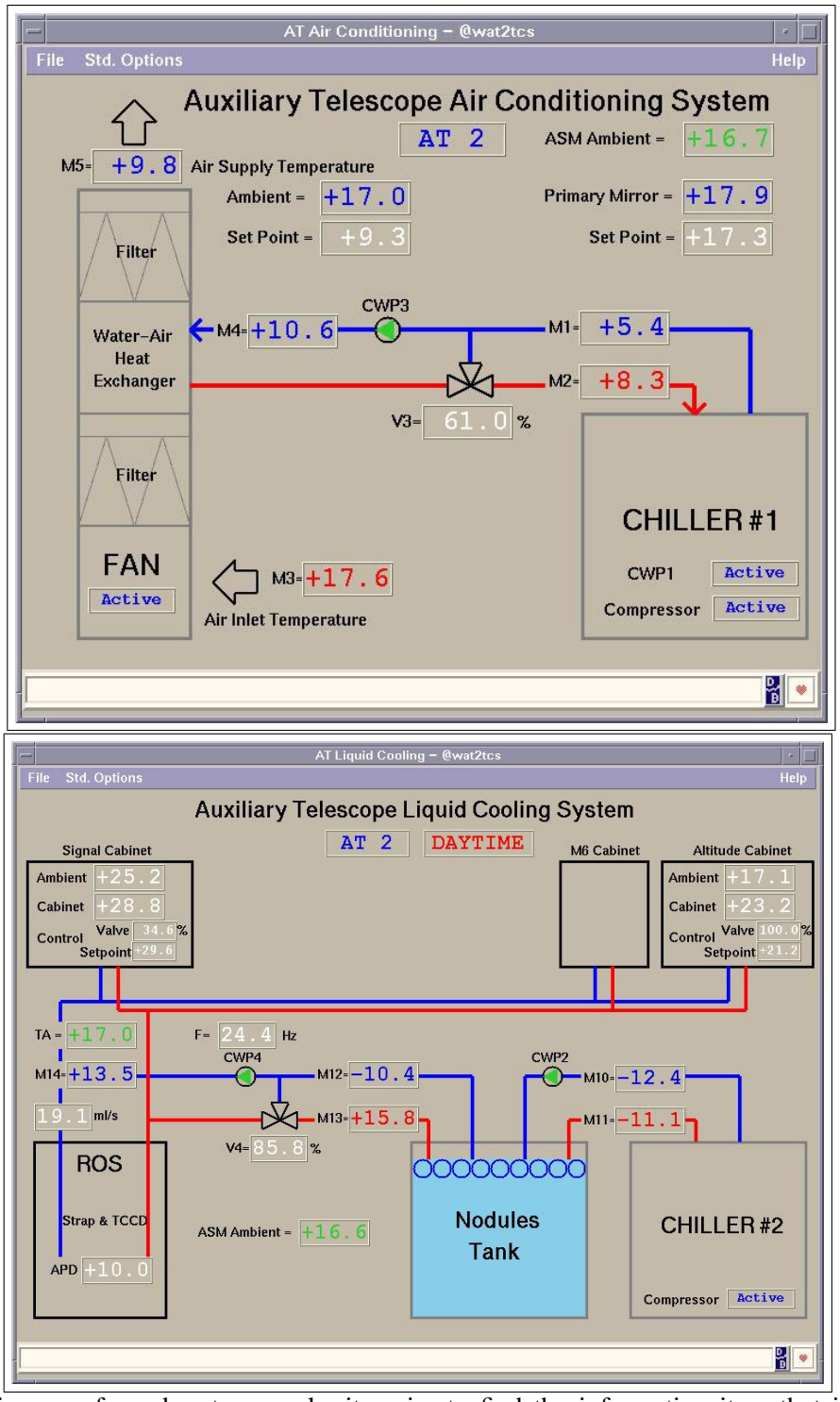


Figure 27: Diagrams for subsystems make it easier to find the information item that is searched for (Paranal/VLT)

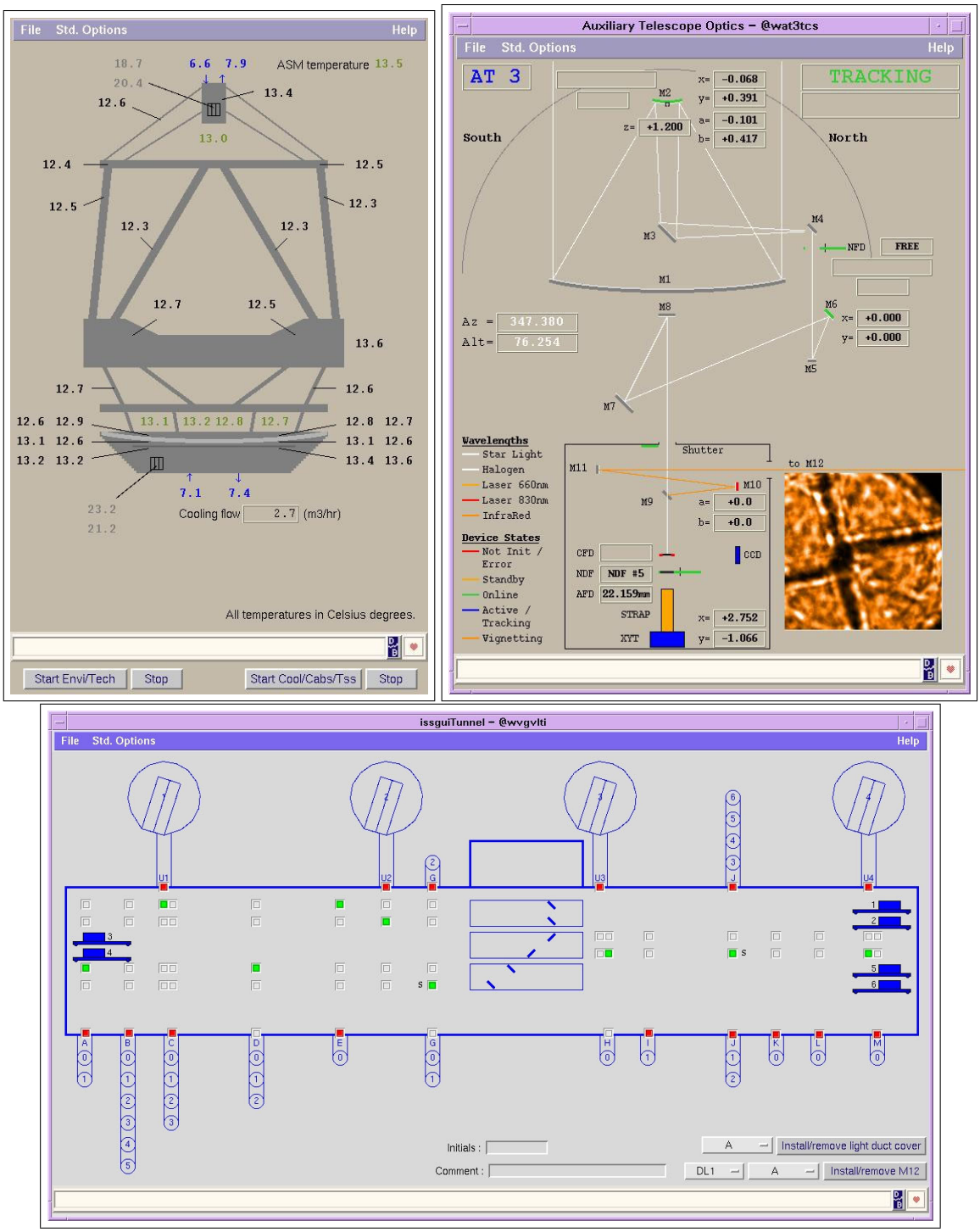


Figure 28: Different examples of schematic representations of devices with status information (Paranal/VLT)

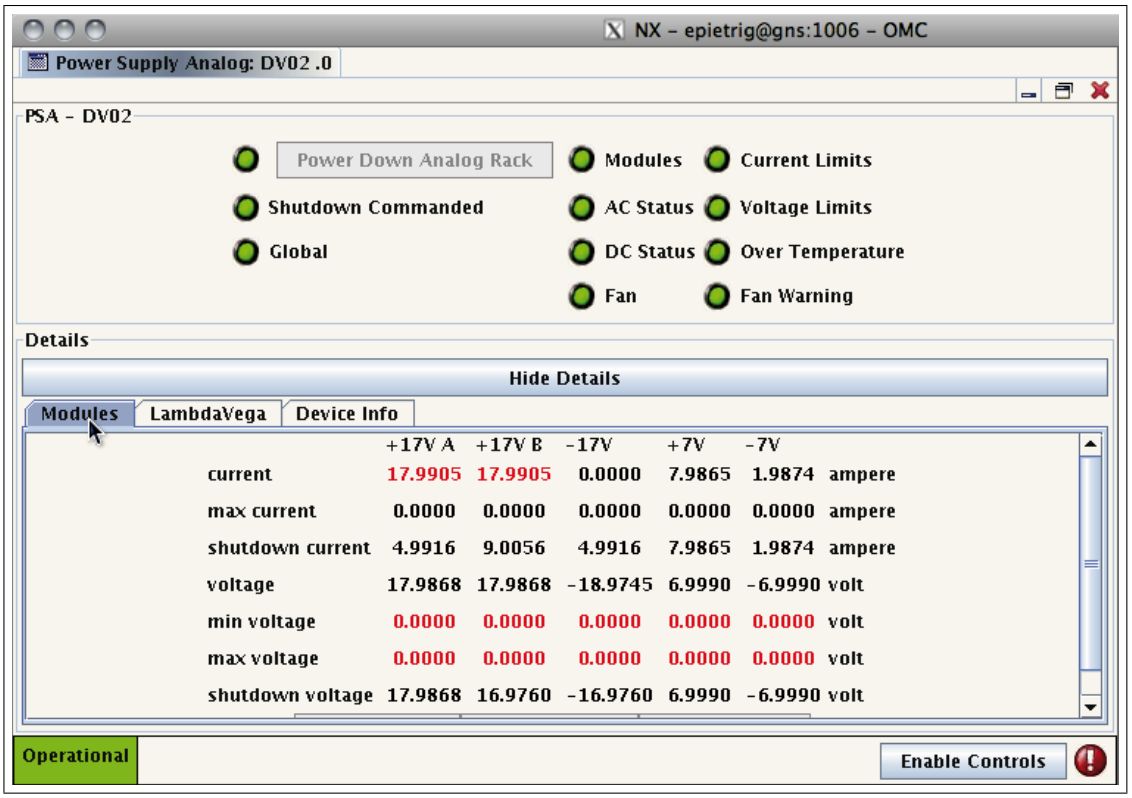


Figure 29: Panel that would typically benefit from the user of a more visual representation of variables

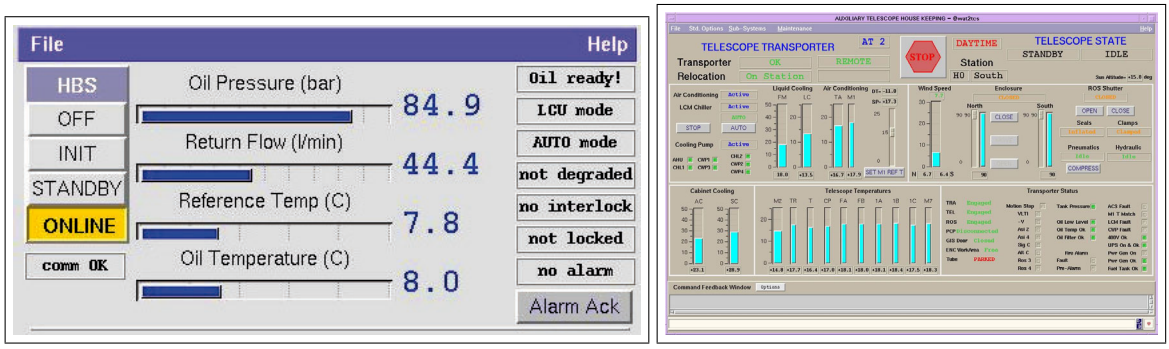


Figure 30: Use of bar gauges to show values that are supposed to vary in specific ranges (Paranal/VLT)

Some panels would also greatly benefit from more visual representations of values. For instance, the `Power Supply Analog` panel features several values that are supposed to remain within a defined range (Figure 29). **S(moderate, high):** Instead of using text-only values, use gauges to display the current value (see examples from VLT in Figure 30) as well as the allowed range on an appropriate scale. The textual representation of values should still be displayed, as the visual representation is not precise enough; its purpose is to provide an efficient mean of assessing the current value and situating it within the range based on pre-attentive variables. It also makes identifying outliers or comparing similar variables much

easier (cognitively much less demanding, as comparing the height of two bars is much easier than reading and comparing two textual values). Beyond the case of the `Power Supply Analog` panel shown in Figure 29, it is stated in [28] Section 3.3.2 Detail Area that *Valid ranges [...] are available by moving the mouse over the value. This will draw a pop-up display showing lower and upper ranges*. The same comment about more visual, gauge-based representations applies. But in this case the problem is worse, as the valid range is shown only in a pop-up; this implies no persistency, difficult to make comparisons, ... as for tool-tips – see comments about the inappropriateness of tooltips to convey information other than contextual help in Section 4.7.

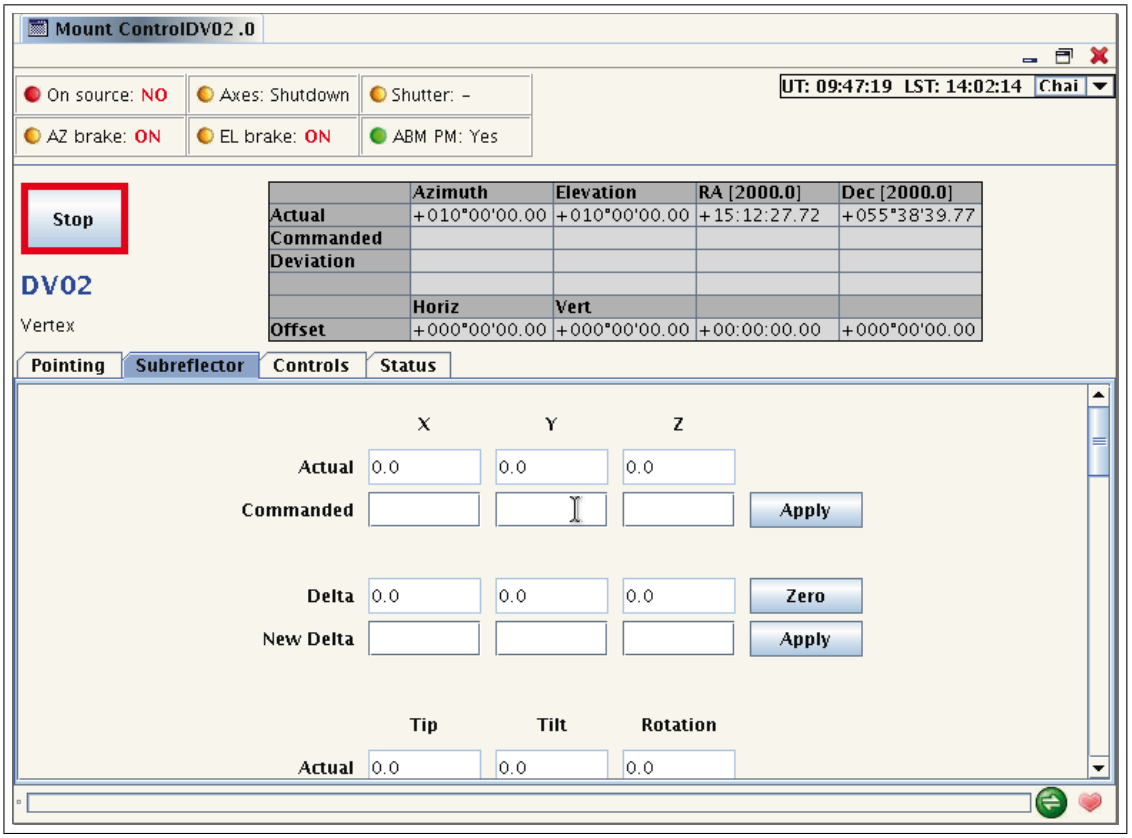


Figure 31:

In a similar manner, some panels show the current value of a variable, as well as the target/commanded value (Figure 31). It is important to get the exact numerical values, but again, visually illustrating how far the current value is from the targeted value, in other words, showing some type of visual progress indicator, would be useful for the same reasons as above.

It is not clear why both the `Mount Status` panel and the `Mount Control` panel exist, as the former seems to be a read-only subset of the latter (Figure 32). **S(easy to moderate, moderate):** Wouldn't it be possible to have a single panel, read-only by default, which could be unlocked, enabling the user to control it (provided the user has the permission for that)? This would decrease the number of plugins in the interface.

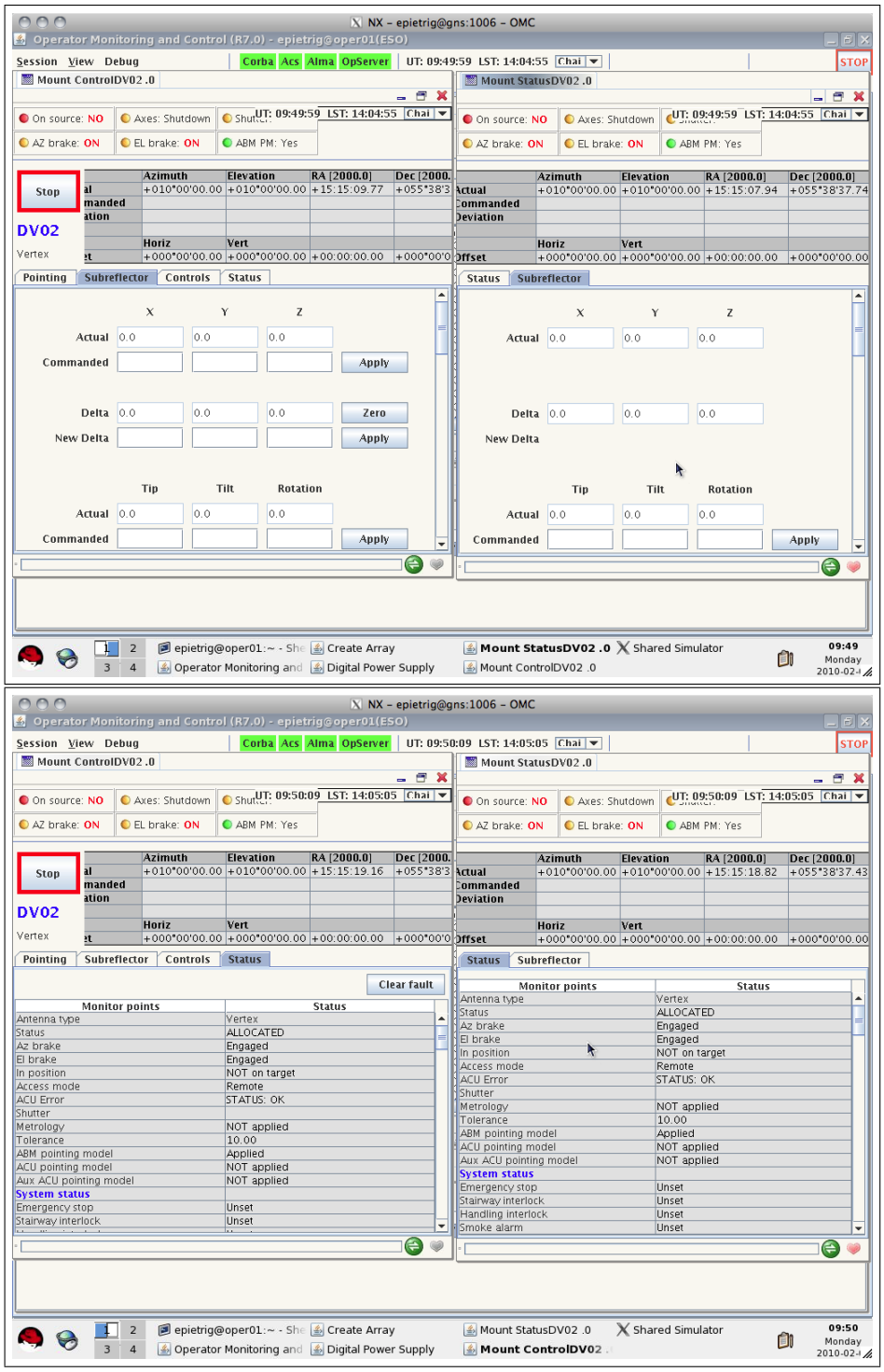
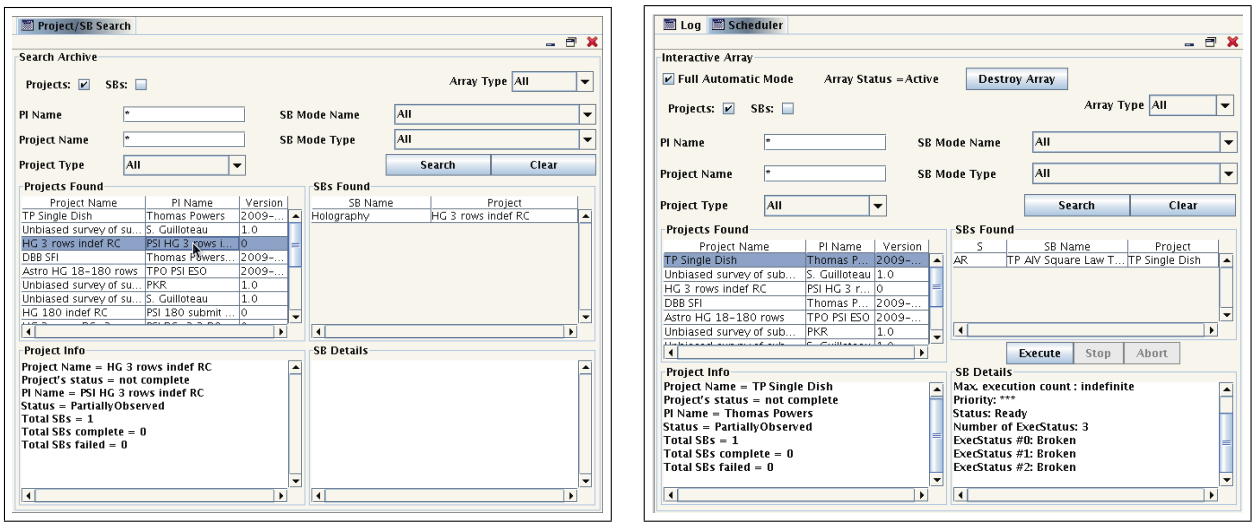


Figure 32: Mount Status vs. Mount Control

4.6 Scheduler, Project/SB Search, Dataflow

Plugins `Project/SB Search` and `Scheduler` have a similar layout, thus preserving consistency across closely-related components (Figure 33). According to [20] the `Search` plugin *"will hopefully be replaced by an archive plug-in search tool"* and *"currently exists purely for convenience"*. The following comments are thus focused on the `Scheduler` plugin, though many of them apply to the former due to their similarity in terms of UI design.



(a) (b)

Figure 33: Project/SB Search panel, Scheduler panel in interactive array mode.

The UI is densely packed, in both Interactive and Queued scheduling modes, giving a false impression of complexity. This is symptomatic of the "quest for screen real-estate" that results from the use of a purely WIMP-based UI design, as discussed earlier (Section 4.1). One direct negative consequence is that the four tables/text areas in the lower part of the pane have to be wrapped in scroll panes that feature both a horizontal and vertical bars. Combined with the small viewport size, this makes it very difficult to read the data displayed in these components, as it entails many scrollbar manipulation actions, but also because the scrollbar widgets themselves take screen-real-estate from an already very small viewport. **S(moderate, high):** redesign this panel so as to avoid horizontal scrollbars. I believe you can afford to make this panel bigger, as it seems to be one that is used often during operations. Obviously, addressing the more general issue discussed in Section 4.1 would make it possible to dedicate more screen real-estate to the Scheduler plug-in.

`Execute`, `Stop` and `Abort` buttons seem to get automatically enabled/disabled depending on the context, e.g., whether an SB is selected in the `SBs Found` table or not. **S(easy, moderate):** make sure that those buttons are enabled/disabled in a consistent manner. Otherwise, users might think that the system (or the UI) has become unresponsive.

Buttons for managing SBs in Queued scheduling mode: the labels do not properly convey the semantics of the actions (detailed in [20]). For instance, `StopSB` and `AbortSB` not only stop the current SB but start executing the next one. This is what makes the difference between `*SB` buttons and `*Queue` buttons. But this is not explicit. **S(easy, low):** Obviously the button labels cannot be made too long. Possible

alternative: resort to icons, that could convey the semantics in a more visually compact manner (it is often said that *"a picture is worth a thousand words"*; though this is not always true, this is a typical case where it could apply). In any case, make sure explanatory tooltips are available for each button.

Status of a scheduling block's execution: the status of an SB is represented by a letter (`R, C, AB, S, AR, F`) in column `S`. **S(easy, low):** relating these letters to actual status is cognitively demanding. Again, icons might be more self-explanatory. And in any case, make sure tooltips are available for each status.

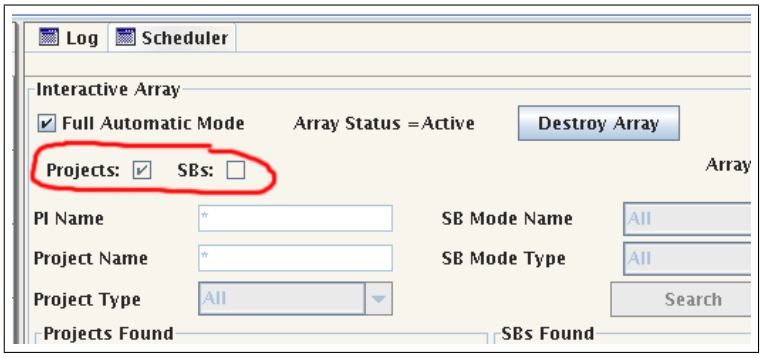


Figure 34: Incorrect use of checkboxes for mutually-exclusive choices

Results by project or by SB (Figure 34): the two modes are available and mutually exclusive. Checkboxes are used for choosing multiple options in a set; when those options are mutually exclusive, radio buttons should be used [33]. **S(easy, moderate):** Simply replace the two checkboxes by two radio buttons.

Searching the archive: right now, search is a very sequential process: users enter query parameters in different fields, click search and get the results. As mentioned earlier, e.g., for JLog, **S(moderate, moderate):** a more interactive search paradigm, such as dynamic queries [32], or even basic suggestions as is now commonly found in Web query interfaces, would make the search more efficient. It is my understanding that the archive will be very large. In this context, dynamic queries would clearly improve search efficiency, but might also be challenging to implement.

4.7 Quick Look

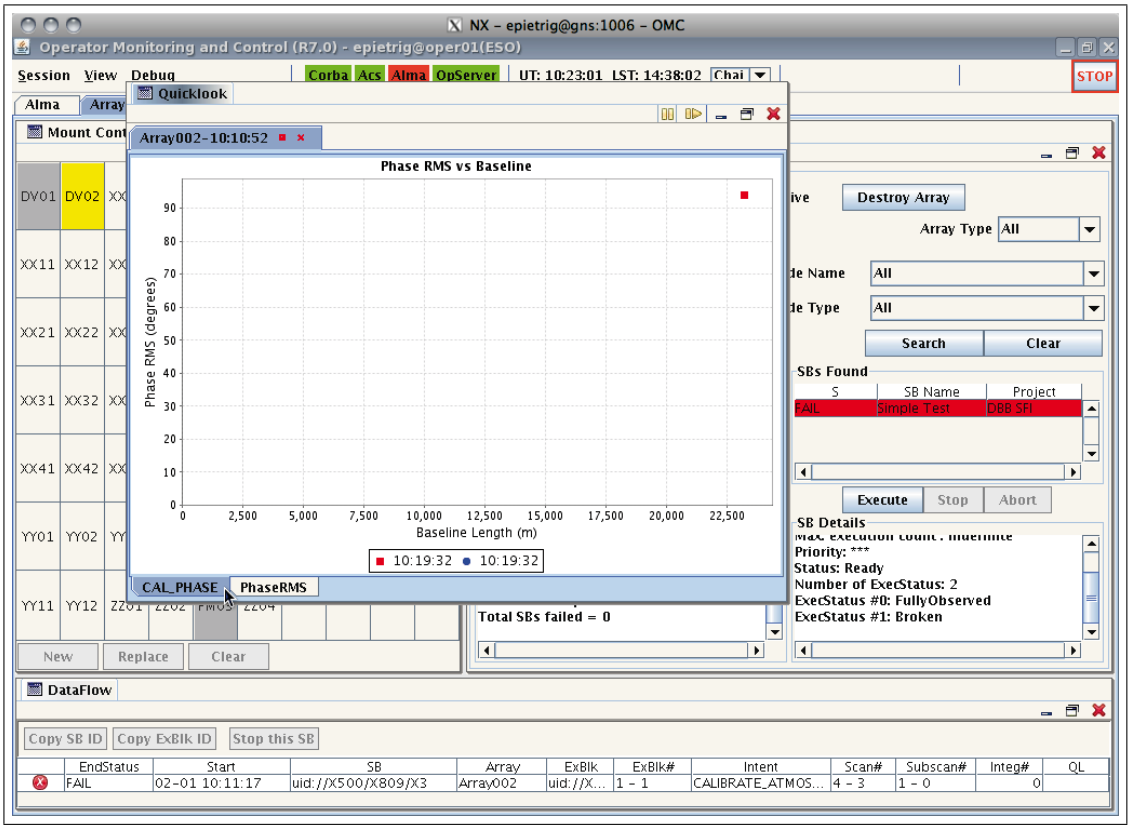


Figure 35: QuickLook running as an OMC plugin

QuickLook can be run as a plug-in or standalone. Calibration results consist of many charts that use different types of plots depending on the nature of the information visualized (e.g., aperture efficiency vs. time, pointing offsets, phase rms vs. baseline length). According to [15], all these charts (13 different types total) can be presented either in `Gather` mode (Figure 37-left) or `Scatter` mode (Figure 36). The former uses less screen real-estate, the latter allows for the simultaneous visualization of several plots.

`Gather` mode is compact and does not require the user to manage many windows. However, it does not allow monitoring several plots at the same time. The user has to click on a tab to see the plot associated with this data, which replaces the previously selected one.

`Scatter` mode assigns one plot per window. Each window can be freely moved and resized by the user. This mode allows monitoring many plots at the same time, but uses much more screen-real-estate (typically an entire screen – Figure 36). It is not clear, reading the documentation [15], how the charts are laid out by default when switching to `Scatter` mode. It is written that they are *"[laid] out sequentially from left side of op console."*. Does this mean that the windows get arranged in a matrix-like, space-filling layout as shown on Figure 36? **S(easy, high):** make sure this is the case, and that the default layout algorithm uses available screen real-estate in an optimal manner. It is good that users can freely move and resize each window in `Scatter` mode, but users should be spared the tedious task of having to manually move and

resize the windows to obtain a usable layout. **S(easy, high):** at the very minimum it should be possible to save/restore `Scatter` mode layouts; additionally, the default layout should be predictable (meaning that the ordered sequence of displays should always be the same, so that a given display always ends up in the same place in the matrix).

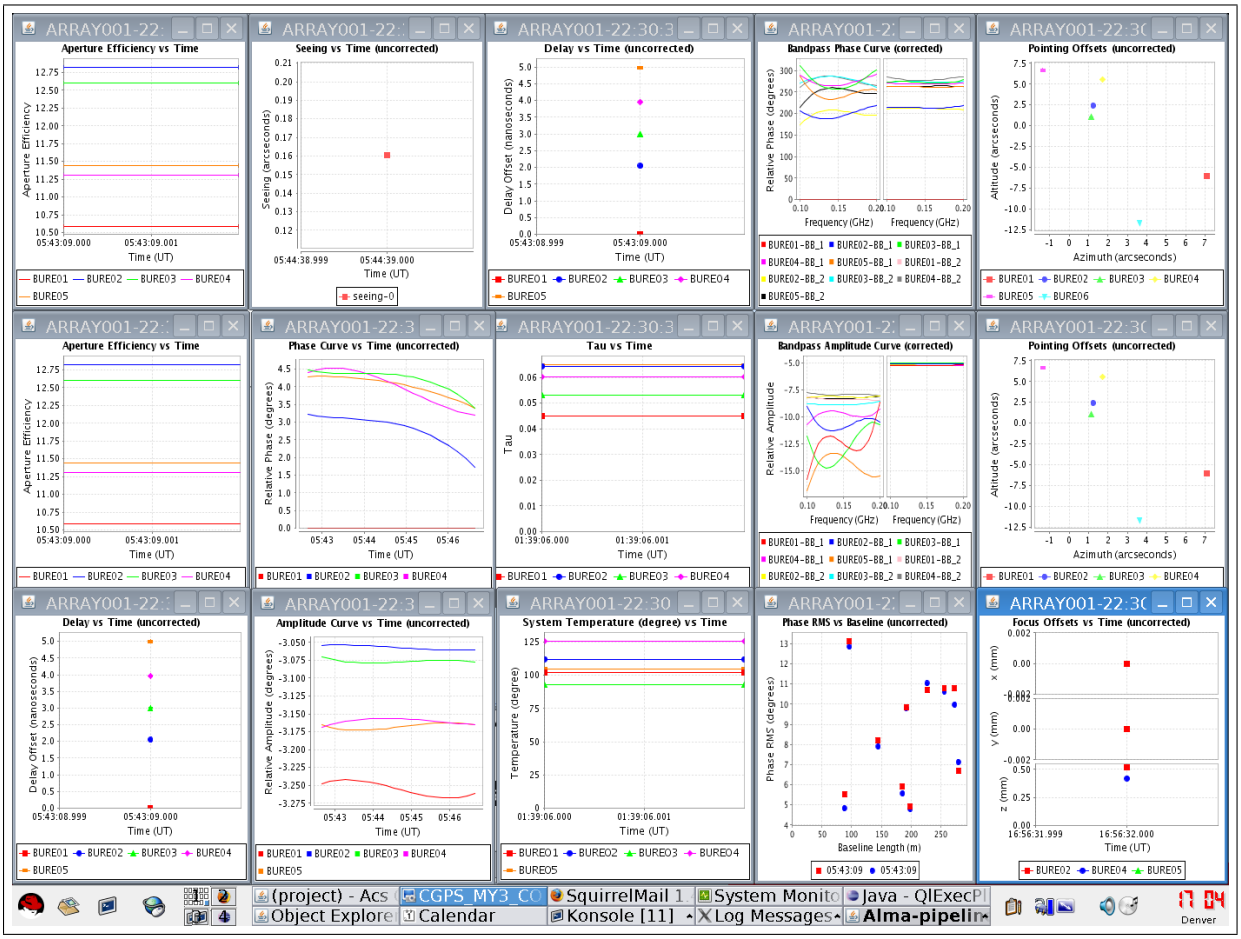

Aperture Efficiency vs Time
Seeing vs Time (uncorrected)
Delay vs Time (uncorrected)
Bandpass Phase Curve (corrected)
Pointing Offsets (uncorrected)
Aperture Efficiency vs Time
Phase Curve vs Time (uncorrected)
Tau vs Time
Bandpass Amplitude Curve (corrected)
Pointing Offsets (uncorrected)
Delay vs Time (uncorrected)
Amplitude Curve vs Time (uncorrected)
System Temperature (degree) vs Time
Phase RMS vs Baseline (uncorrected)
Focus Offsets vs Time (uncorrected)

Figure 36: Problems of scalability in terms of number and complexity of the data plots to monitor

Beyond the problem of having to monitor and manage more than a dozen different charts, the major problem of QuickLook is that the current plotting solutions will hardly scale to 60+ antennas, and will definitely not when thousands of baselines have to be represented. The main issues are related to the legibility of the plots:

- It is not possible to overlay in the same window more than 60 plots (except for scatterplots).
- It is not possible for users to distinguish more than a thousand different color-coded baselines.
- It is challenging, and for most people impossible, to efficiently distinguish more than 60 color-coded antennas (it would be challenging even if there were just 20 or 30 antennas).

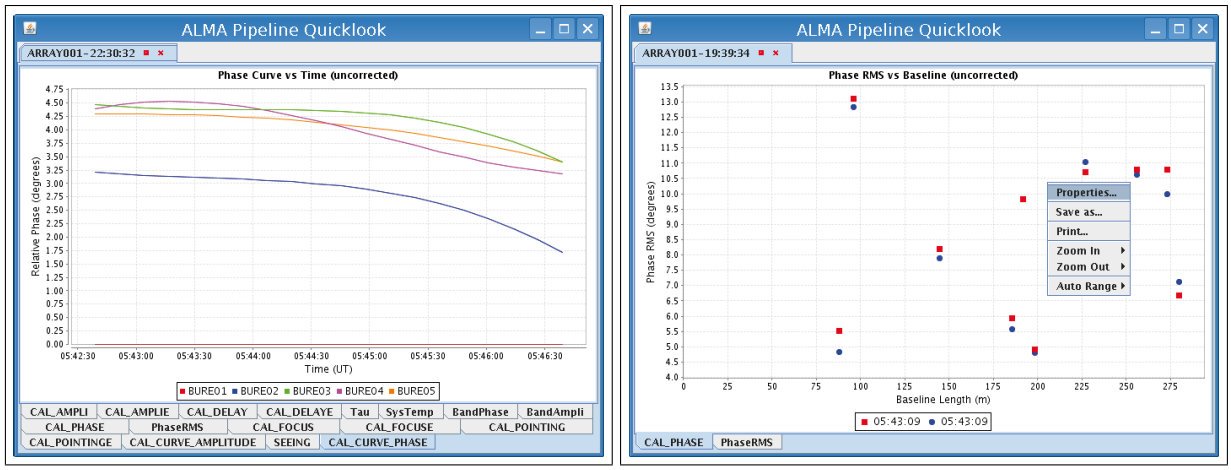


Figure 37: Problems of scalability in terms of number of antennas and baselines. Conventional charts and plots work well with a few antennas/baselines, but will not scale

S(difficult, major): There is no quick and easy solution to this problem. Addressing the scalability issue requires more UI design work, working closely with end users (operators and astronomers) to understand how these plots will be used (what information is critical? how often do users look at them? what actions are they likely to take in response to problems identified in the plots? what is the time frame/delay before these actions get reflected in the plots, if at all? do users need to establish correlations between events in different plots? etc.). Ideas that have crossed my mind include:

- For a given plot, only present a subset of all antennas/baselines at a time (say, maximum 10 of them[3]), and automatically rotate between subsets every N seconds. The user must however be able to take control of this rotation, e.g., through a temporal slider that would allow her to jump to a given subset quickly.
- As the temporal slider (operating on a continuous scale) approaches the range dedicated to another subset, gradually fade-in plots from the next subset, and gradually fade-out plots from the current one. This can easily be done with simple alpha blending (`src_over` compositing); balance between alpha values in the transition range would have to be set through iterative refinement. The goal here is not to produce an aesthetically pleasing result (in a way, this is a side-effect), but to allow users to compare and contrast plots from consecutive subsets.
- For a given time-series, if the Y-value for a given time stamp is supposed to be about the same for all antennas/baselines (in other words, if all plots are supposed to be superimposed or very close to one another), then no matter which subset is currently displayed, show some kind of low-contrasted gray background filled envelope bounding minimum and maximum Y-values across time. This will help identify outliers.
- Possibly add a 3D plot to the chart (orthographic projection) that will serve as an interactive overview of all antenna/baseline plots. X and Y axes are the same as in the main chart, Z axis is used to

[3]The optimal number will probably depend on each chart, knowing that it might actually be better to choose the same not-necessarily-optimal number for each chart, so as to maintain consistency between them.

organize the 60+ or 1000+ single plots into a 3D surface, that shows an overview of all values (not much detail, but all plots seen at the same time) and can be used as an alternative to the temporal slider to select which subset to display by direct manipulation.

The above ideas are just examples of possible but partial[4] solutions. They could be combined with other research results in time-series visualization, e.g., [13, 14, 18], and would have to be discussed, refined and tested with users.

S(moderate, moderate): The following are easier-to-implement suggestions for improvements to the current UI. They are clearly incremental and do not address the above issues on their own:

- Selecting a plot by clicking on its name in the caption area should highlight it in the plot area (wider stroke, brighter color, ..., but no blinking). Conversely, hovering a given plot should highlight its name in the caption area (and trigger the same highlighting for easier reading and consistency of behavior).
- To allow easier comparison/contrasting and to make it easier to check for correlations, add synchronized vertical bar cursors to the different charts that share the same X-axis, e.g., all time series, as the currently active chart (the chart in which the mouse cursor is currently situated). The same can be done with a horizontal cursor for the Y-axis when it makes sense.
- According to [15], *"A drag and release in the direction towards lower-right will zoom in. A drag and release in the direction towards upper-left will zoom out (to restore the unzoomed)."* This is somewhat unconventional and not very intuitive. Moreover, it is not clear how this fits with panning in the graph when zoomed-in (which is usually performed with left mouse button dragging). The mouse wheel should be supported for zoom in/out, and panning should be performed with the left mouse button.
- According to [15], left mouse button clicks anywhere in the plot toggle between the display of corrected and uncorrected data. This hidden, not-explicitly-represented action is potentially dangerous. In addition, using the color of the chart's title to indicate the current display (corrected vs. uncorrected) is not advised. It would be better to have a small button labeled with the current status, that toggles between both displays when clicked.
- Tool tips to display information about data points [15]: tool tips are usually used as a means to convey helpful information to the user. They thus appear after some delay once the cursor has stopped moving. This behavior is appropriate in the context of help, but not when the goal is to convey information to the user. The delay decreases efficiency and increases frustration. Make sure the tooltip appears immediately. This will however not solve the problem of persistence: tooltips disappear as the mouse cursor is moved, making it difficult for users to compare values for different data points.
- Selection of data points: if users are expected to select data points often, a pointing facilitation technique (making it easier to select small targets) could be used in the chart area, such as the Bubble Cursor [11] or DynaSpot [5]. Another option would be to use excentric labels [8].

[4]Partial because they do not allow some operations that might be essential, such as comparing and contrasting plots for antennas or baselines that belong to different subsets.

4.8 Miscellaneous

4.8.1 Animations, Widgets, Layout, Colors and Fonts

Figure 38: Part of the OMC status bar

Colors are used in many places in the UI to reflect the status of hardware/software components, using green, orange/yellow, and red. This conventional color coding scheme is appropriate, but because of the fairly intensive use of these color throughout the interface, fine-tuning the colors is important as too bright/vivid color values can be tiring for the human eye. Knowing that operators and AoD will watch the display for long hours, this issue should not be neglected. **S(easy, high):** desaturate or darken the colors, and pay attention to color blindness (it was anticipated that operators would not be color-blind, but actually, we met at least three people in Santiago and at the OSF who were color-blind). Note: the colors were very saturated and bright when we were at the OSF (see Figure 39), especially the red and green. The green I saw when exploring the OMC in February seemed darker (Figure 38), but the red is probably still too bright (Figure 39 bottom-left).

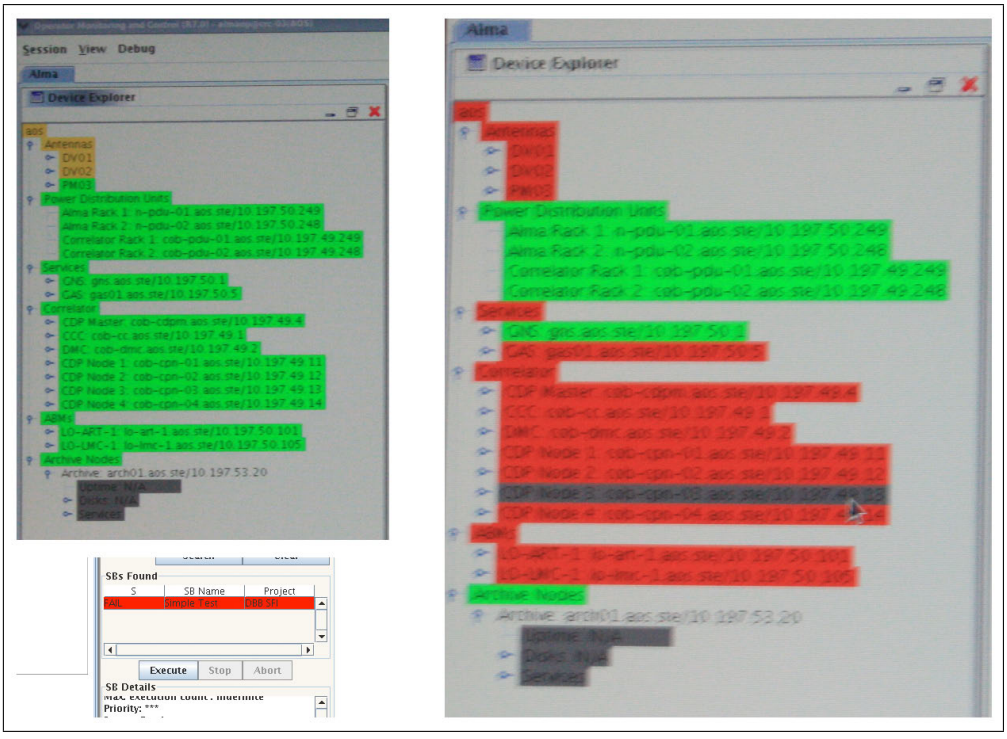


Figure 39: Highly-saturated colors are tiring for the human eye

As for several VLT panels (Figure 40-right), several ALMA panels feature a small blinking heart (Figure 40-left) to indicate the status of the software UI panel's connection with the actual system component. Blinking elements are very effective at drawing users' attention. The current behavior (blink when connection is OK, thus most of the time) is thus not advised. **S(easy, low):** Blinking should be used for short periods of time, to draw the user's attention when something is wrong. Not the other way around.

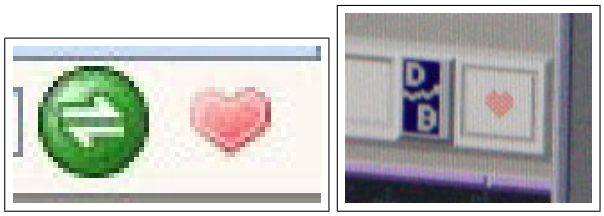


Figure 40: Blinking heart in the OMC interface (left) and at the VLT (right)

The current drop-down menu lists all plugins in an uncategorized flat list (Figure 41). **S(easy, moderate):** Grouping plugins by category, e.g., all control device UI panels, in a hierarchical menu would help navigate in the set of available views.

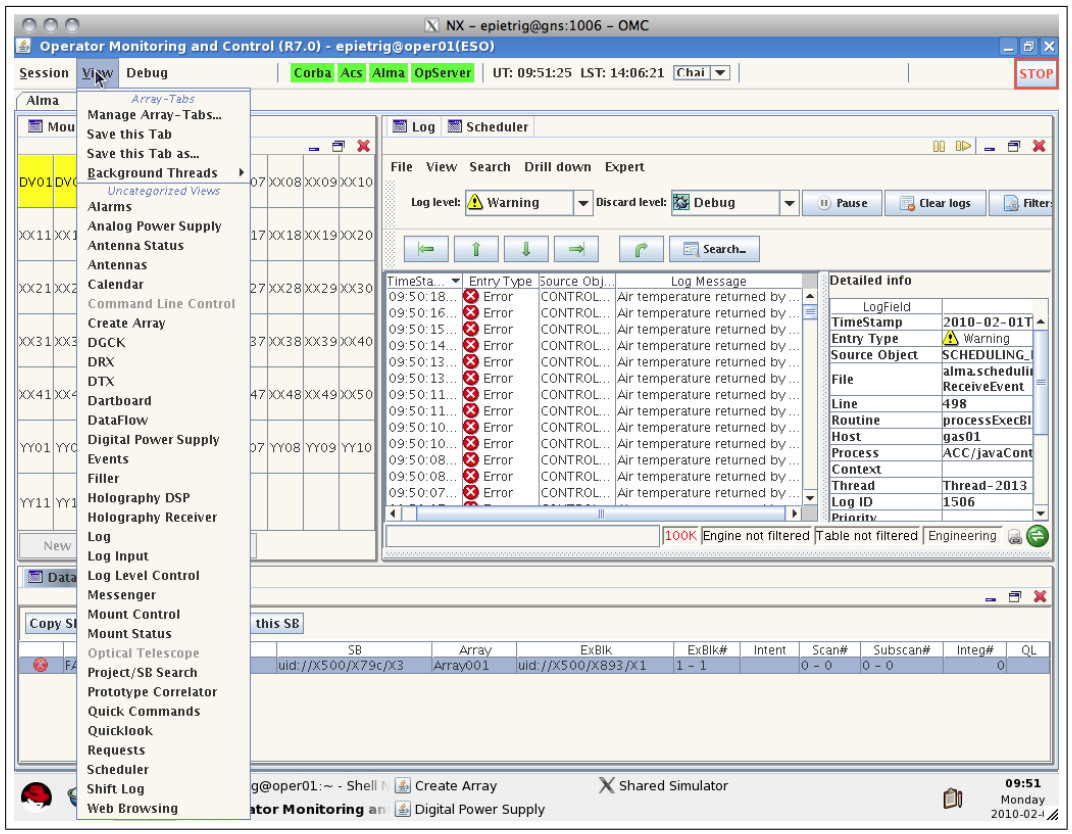


Figure 41: Uncategorized views make for a log flat list that is tedious to scan

S(easy, moderate): Finally, because of the relatively low resolution (DPI) of current LCD panels, italic fonts should be avoided (Figure 42), as they are more difficult to read than regular fonts. The use of italic fonts is also mentioned in Section 6.4.1 ACS Explorer Panel of [10]. The information should be conveyed through another means.

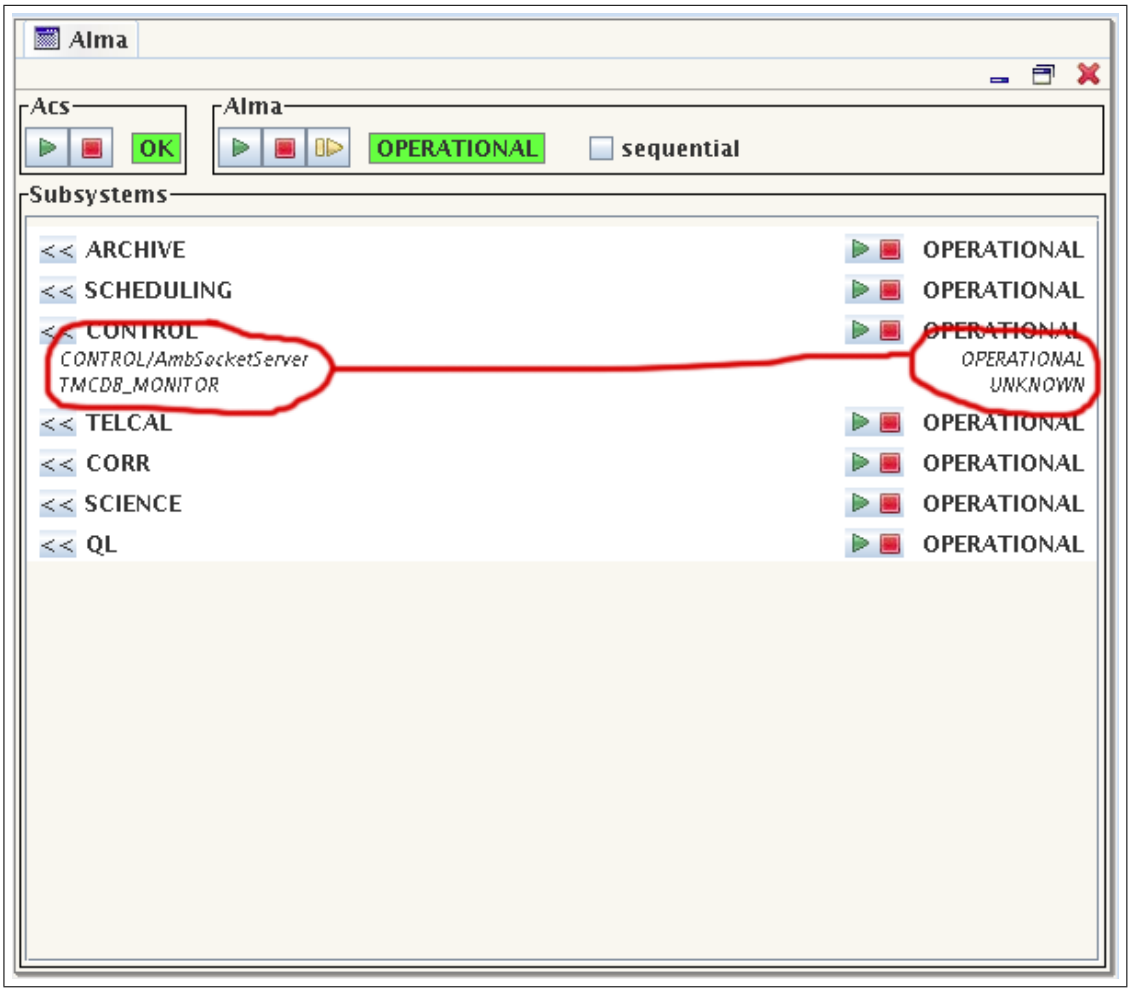


Figure 42: Italic Fonts are legible on paper but perform poorly when displayed on screen

S(easy, moderate): Double clicks should not be used for simple selection actions. For instance, [10] Section 6.7 states that events (rows) in the Calendar panel have to be double-clicked to get details displayed in the right-most section. This is not consistent with conventional interaction with rows in tables. A single click should be sufficient to display detailed information (unless retrieving the details is time/resource consuming and there are scenarii where one might want to select a particular row without seeing the details).

Some panels feature a high level of component nesting, e.g., Figure 43., making it more difficult for users to orient themselves in the UI, as tabs containing many border panels are nested within other tabs that are themselves presented in panels that can be collapsed, that are themselves part of a panel within the

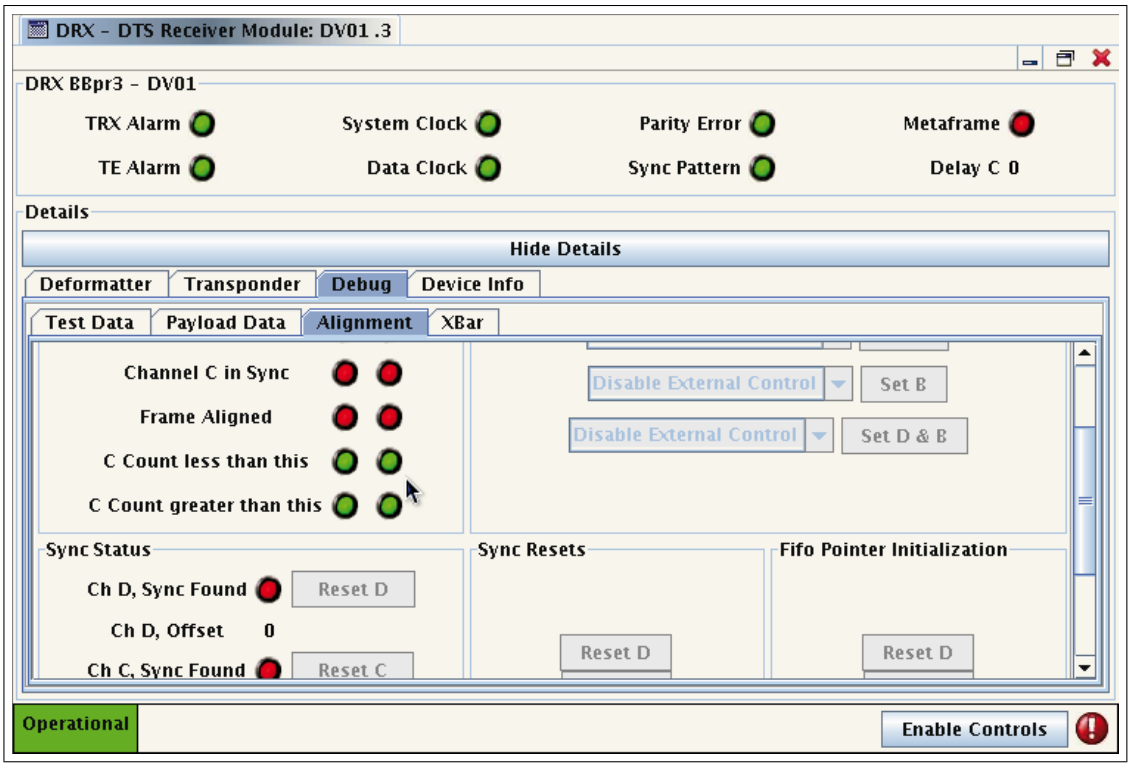


Figure 43: Excessive nesting of components makes it difficult for users to orient themselves in the interface

docking framework... Avoid nesting of tabbed panes, **S(moderate, moderate):** either by using other Swing widgets to organize the interface, **S(easy, high):** or naturally, as part of the switch to a multi-scale interface as suggested in Section 4.1 (but the latter is **S(difficult, high)**).

Several pop up menus feature a `Cancel` item, that seems to be used only to discard the menu. **S(easy, low):** Usually pop up menus are discarded simply by clicking outside the menu. The semantics of this `Cancel` item are thus ambiguous. Removing it will also shorten the list of items, and hence the menu's complexity. **S(easy, low):** Add separator lines to group menu items logically, even if the menu does not contain many elements. This will reduce errors (accidental selections) and make visual scanning easier. **S(moderate to hard, moderate):** Generally speaking marking menus [19] or even just pie menus would be more efficient than linear pop up menus.

4.8.2 Web Browsing

According to [29], a Web browser should be available on operator/AoD workstations to *"bring up ALMA tools on the web (e.g. long term schedule information, project status information, problem reports)."*. In this document an external browser is used as an example. It is not clear why a limited internal Web browser is made available as a plugin in the current implementation of the OMC, instead of using a full-fledged external Web browser. The internal Web browser, probably implemented on top of an existing Java-based HTML+CSS renderer, clearly lacks many Web navigation features, fails to render even simple pages correctly (see Figure 44), and contributes to making the list of plugins longer, for no apparent reason. **S(easy, moderate):** An external browser should be preferred, unless there is a clear restriction that forbids conventional Web browsers from running on operator/AoD workstations.

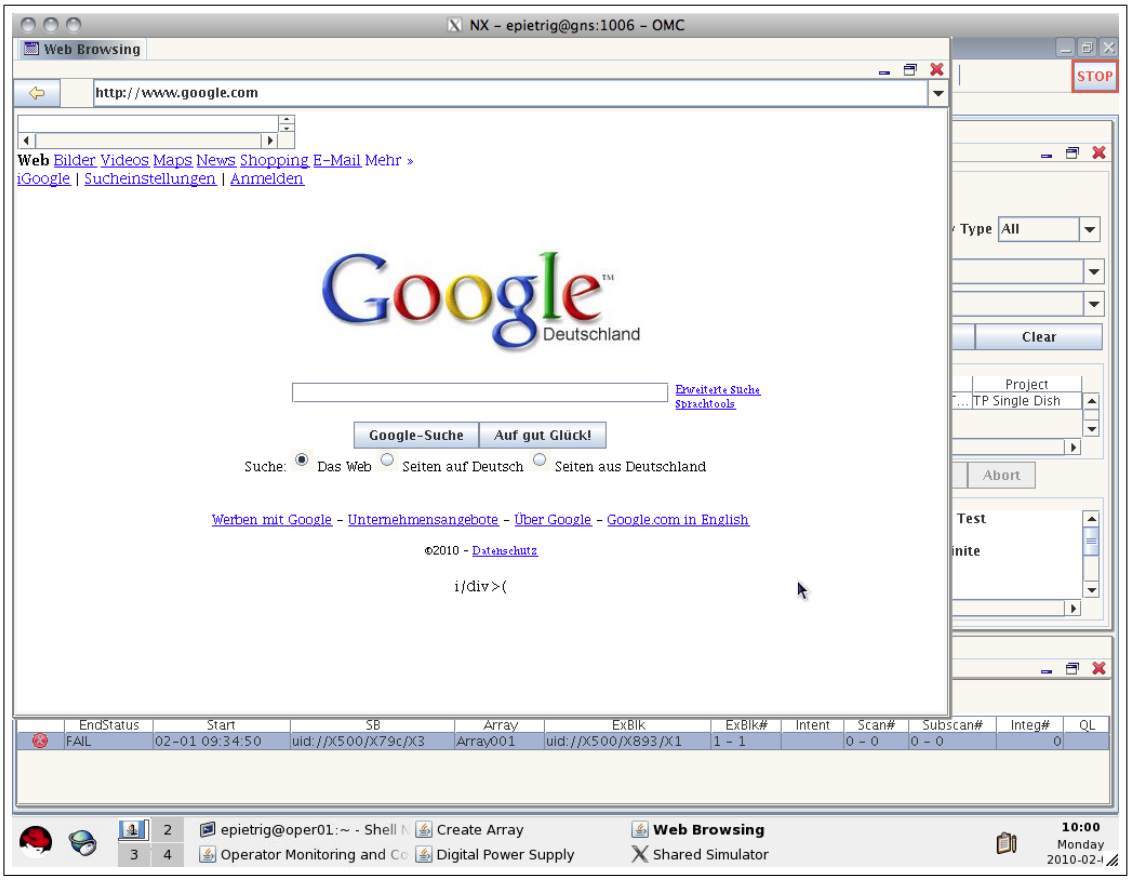


Figure 44: Web browser plugin for the OMC

5. Control room

Contact author : P. Cubaud (cubaud@cnam.fr)

5.1. Control room layout

Control room organisation can be described as an "unthought-of" for the ALMA program. While scanning through the available documentation online, we have not been able to find a specific document (or a chapter) dedicated to this subject. Such a description is however necessary when a standard top-to-bottom design approach is followed in a project. The overall physical organisation of the control room dictates the operators workspace's organization which in turn impacts on the screens space and interaction design.

On the other hand, ALMA is clearly an experimental system, so it is understandable that many "transient" organisations for the control will be necessary before reaching a steady-state (as was the case for the VLT). For instance, the number of people (operators, astronomers) that are supposed to work together in the control room seems to vary. In [29] a number of four people having each a dedicated station is suggested (for the commissioning). Four roles such as "master operator" (MO), "assistant operator" (AO), "staff astronomer" (SA) and "monitoring agent" (MA) are also mentioned (p. 12). On the other hand, [Software architecture] suggests that only one console is necessary (for normal operation). The role of the control stations at the AOS should also be clarified for our analysis. We shall assume here that four people are supposed to work together.

The only document adressing the issue of physical layout for the control operations seems to be [29, figure 6] that we reproduce below in full. (Was this setup supposed to be at OSF or in front of the antenna array at the AOS? the open window suggests the latter.)

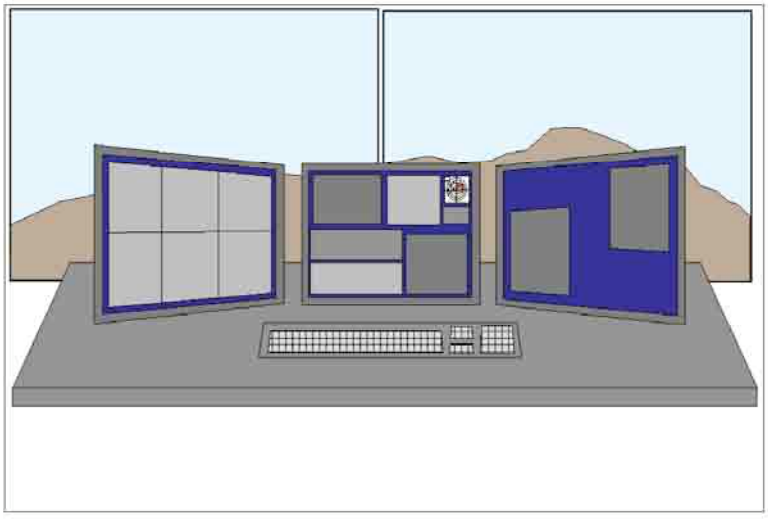
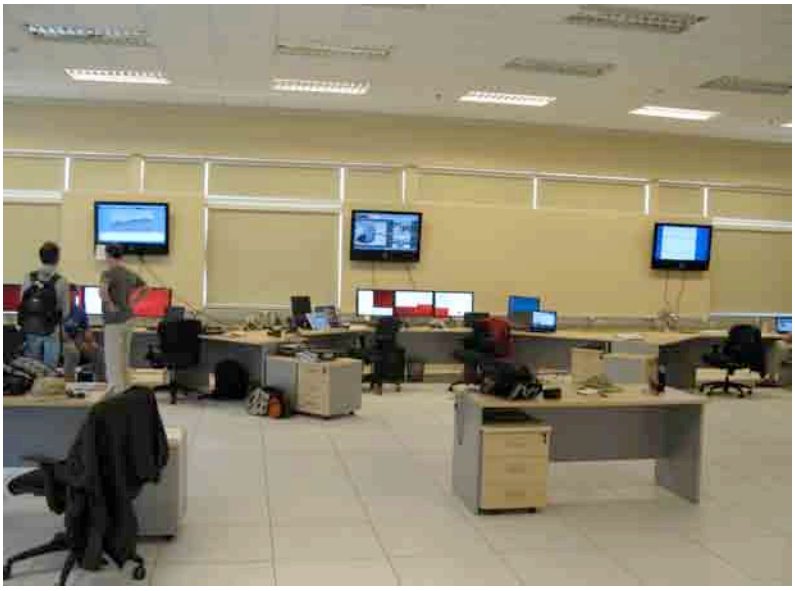


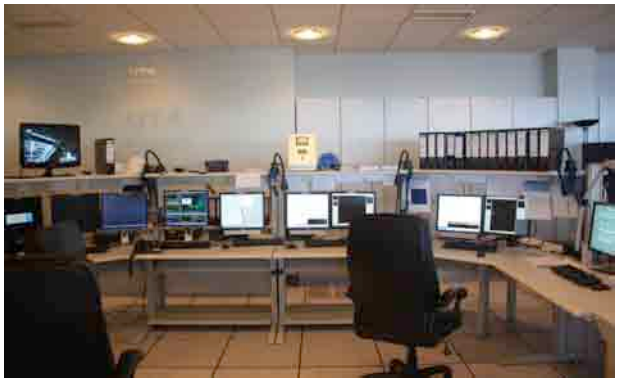
Figure 6: Cartoon of the layout of the operator console configuration. The central monitor displays the array overview and provides a command line interface. The monitor on the left provides detailed status displays. The monitor on the right is open for the operator to start up, e.g. an Observing Tool, Netscape, an Archive query interface, the QuickLook display, or an offline reduction session.

During our visit to the OSF in dec. 2009, the control room was organised as follow. Three consoles were being installed. Room space is big enough to hold a few more. Windows are closed.



Three LCD video displays (TV sets ?) are used to monitor the antenna array. Each operator has 3 screens, as in [29], but a few more will certainly be necessary when all the panels of OMC will be ready (see chapter 4 of this report).

At VLT, almost 6 meters of screen are devoted to operators (For each of the 4 telescopes. Setting for the VLTI is a bit larger). The workspace is organised in horse-shoe. The screens are divided in two equal numbers for the operator and the astronomer on duty.



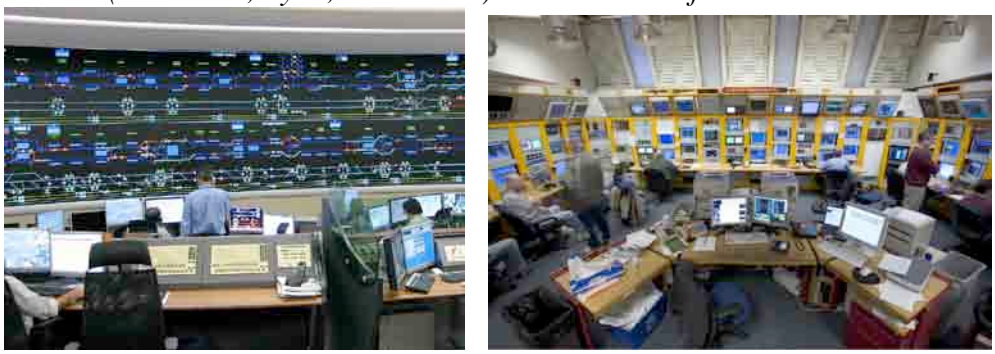
Screens are grouped in pairs with a common keyboard and mouse. There is no pixel space sharing betwen the screens. On the screens below, the partly hidden window on the left side is not a part of the window on the left side.

Each pair of screens is dedicated to a specific task for the operator (initial calibration, etc.) and its location has been chosen accordingly.

Proposal for general layout

Pictures of control rooms abound on the web. All have in common to group general, synthetic information into a "synoptical view" that can be read by all operators. In "analog days", these screens were mostly hardwired with light bulbs, while operator consoles used switches and potentiometers for remote control (see below). Today, many technologies can be used that allow animation and video. Videoprojectors can ensure a large screen projection with good contrast (when equiped with redundant lights, they have a MTBF long enough - cf. Barco website for products).

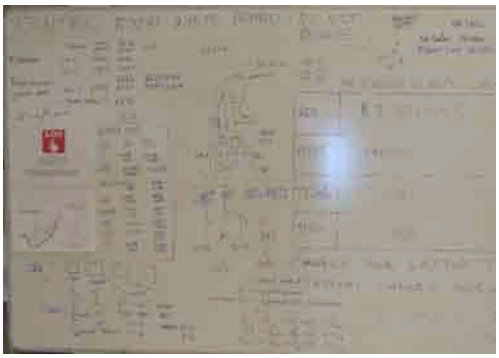
RIGHT : EDF, national electricity network dispatching (circa 1970) LEFT : Motorway tunnel (Fourvière, Lyon, mid 1970's). Note the use of video control.



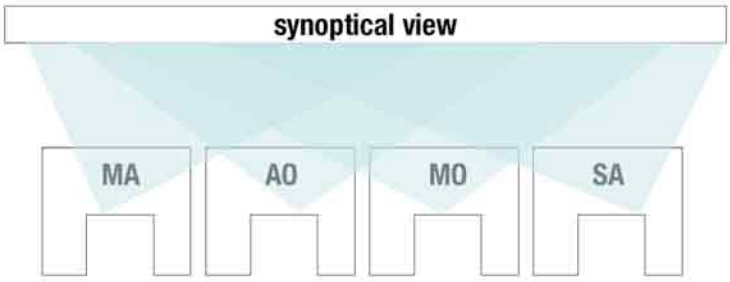
RIGHT : Rio de Janeiro subway (from Barco website, 2010) LEFT : A counter-example : the Tevatron Control room of Fermilab (copied from the website, 2010)

Due to the nature of the process being controled, there is no evident need for a synoptical view at the VLT. Each telescope is operated individually. The collective knowledge (from one telescope team to the other) is reduced to the clock and weather conditions, so screen duplication is not a major issue. This is not the case for ALMA.

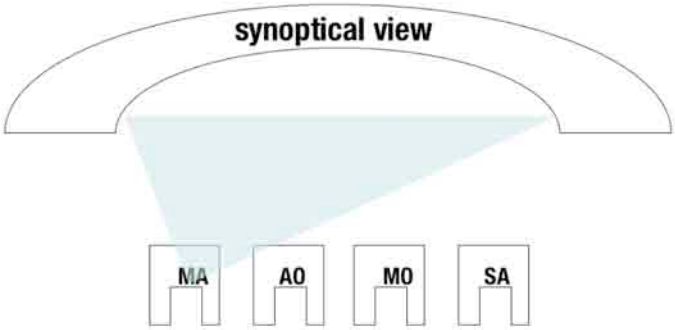
The video monitoring seems a good starting point for designing the control room layout. Video monitoring was initialy unplanned, but many interviewed people acknowledged the need for it. We should recall here that ALMA control is indeed a remote process control. It is understandable that operators need a grasp of the real situation that can be trusted (this should be investigated later by psycho-ergo ?). Another interesting item was the whiteboard found in the control room (picture below), along with schematic posters lying on walls (picture missing).



The control room can be organized in many ways. Using a large rectangular pannel of screens limits the field of view for the various operators :

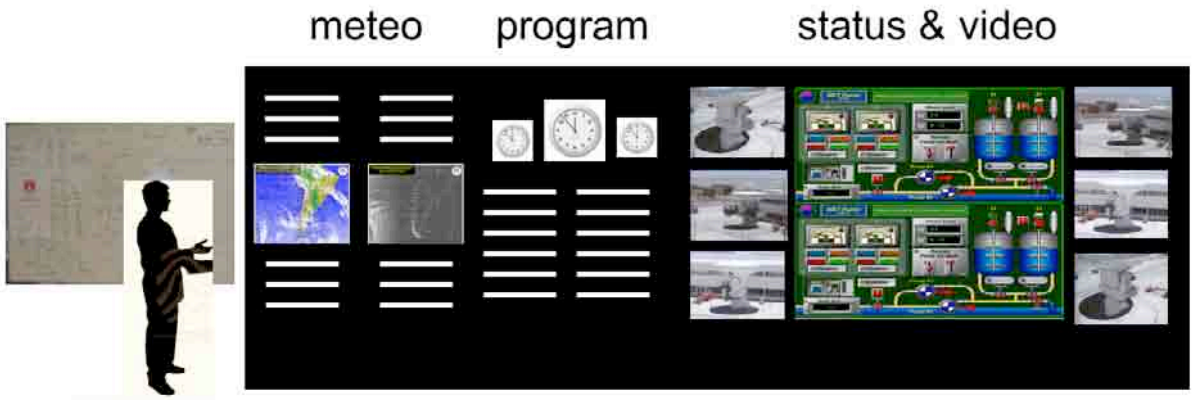


This can be harmless if the people are in front of their most frequently used information. This organisation is interesting if the content of the synoptical view can be shared with people from outside, looking through a glass window (as it is the case at the OSF). Another usual configuration is a circular shape :



Further investigation requires the actual OSF room and operator consoles dimension, but we believe that the actual room is satisfactory.

The synoptical view should include the informations shared by all actors in a synthetic way. This could be :
- Meteo
- The running observation program (list of scheduling blocks and thier current status)
- On-site video monitoring (as it is done already)
- Global state view of the telescope



5.3. Global state view : questions for design

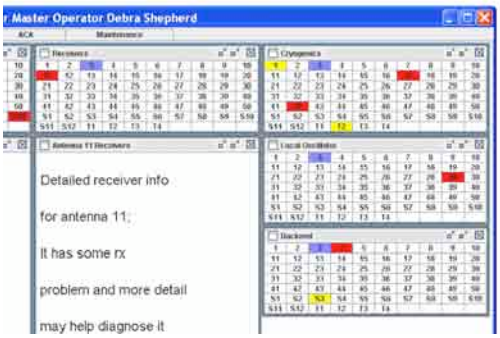
The goal of the "global state view" is to provide a summary of the status of all parts under control, coordinated with a schematics of the process that explicits the relationship between the parts. It is a "read only", permanent, animation, with no interaction with the operators. Its overall readability must be carefully studied.

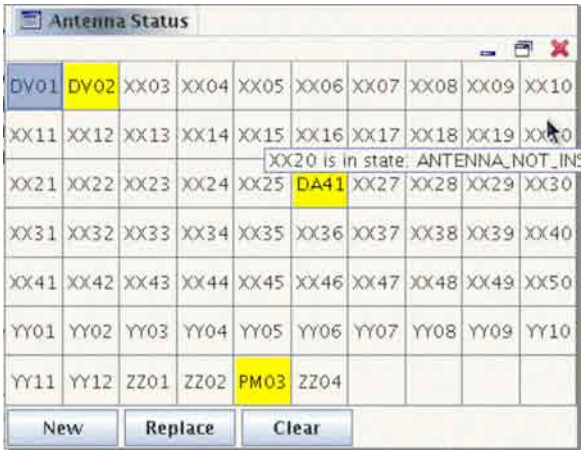
To our knowledge, there is no paper in the ALMA documentation that provides a global view of ALMA in operation, but we have been told during interviews at Santiago that such a schematic was under study. Having this information at hand would very usefull for further HCI investigations. Meanwhile, fig. 4 of [Software architecture, p. 15] can provide guidelines, although it doesn't describe the information flow between the parts. Clearly, the challenge for graphical design here is due to the number of repeated items : 64 antennas, thousends of baselines, etc.

We explore below 3 possible designs.

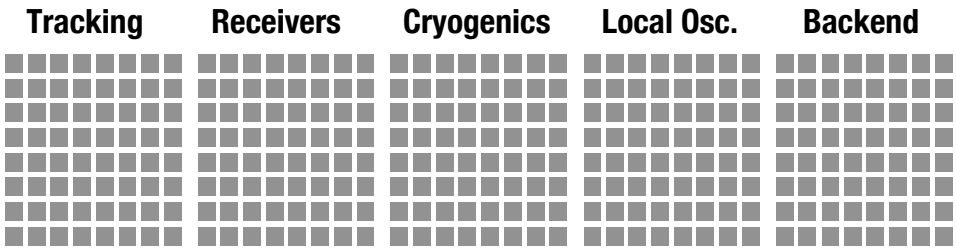
5.2.1 Synthetic approach (chessboard based)

We focus here on the individual antenna status, but are well aware that many other informations have to be reported (calibration phase, execution of scheduling blocks, etc.). Antenna status depends upon 5 subsytems : tracking, receivers, cryogenics, local oscillators and backend. [29] fig. 1 proposes a matrix (10x7) organization. The matrix can be resized at user will. Below are the chessboard from [29] and a screen shot taken at the OSF :



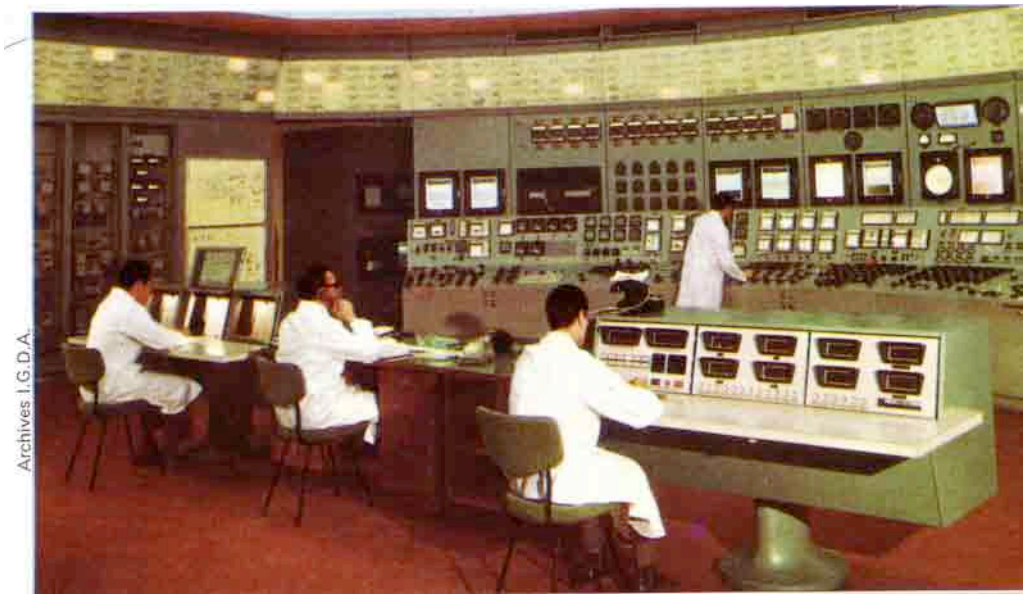

Antenna Status
DV01 DV02 XX03 XX04 XX05 XX06 XX07 XX08 XX09 XX10
XX11 XX12 XX13 XX14 XX15 XX16 XX17 XX18 XX19
XX20 is in state: ANTENNA_NOT_IN
XX21 XX22 XX23 XX24 XX25 DA41 XX27 XX28 XX29 XX30
XX31 XX32 XX33 XX34 XX35 XX36 XX37 XX38 XX39 XX40
XX41 XX42 XX43 XX44 XX45 XX46 XX47 XX48 XX49 XX50
YY01 YY02 YY03 YY04 YY05 YY06 YY07 YY08 YY09 YY10
YY11 YY12 ZZ01 ZZ02 PM03 ZZ04
New
Replace
Clear

A generic 8x8 matrix is used here for simplification. When depicting the subpart states for all antennas, one can use a group of 5 matrices in sequence :

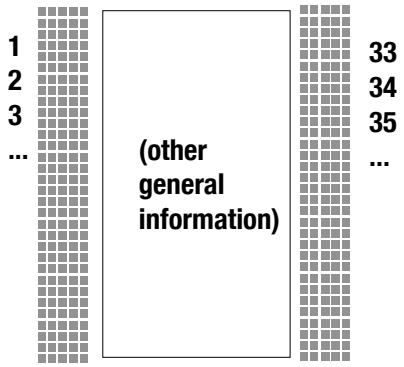


A linear variation :

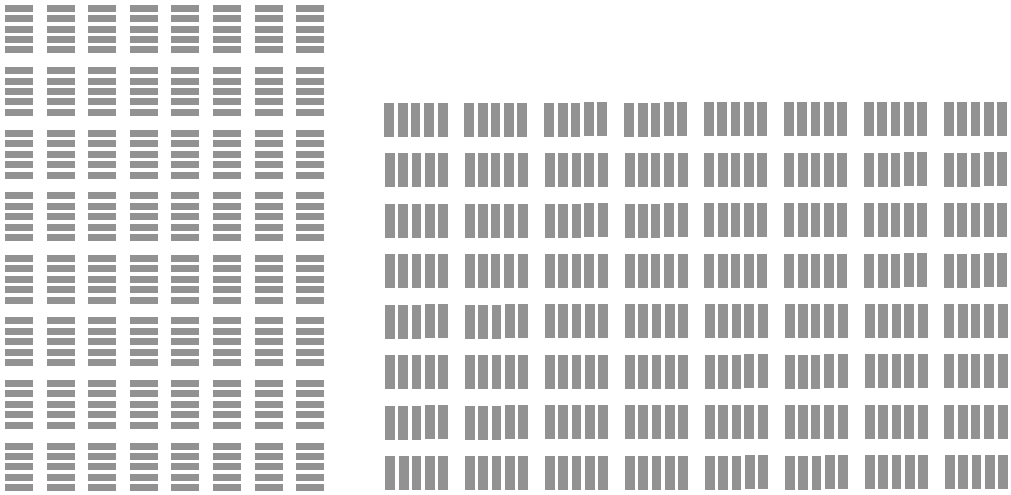
64 x 5 is maybe too big, unless it runs along all the synoptical view, as it is on the top of the picture below :



A possibility is to have it split in two, using the central space for general information of the array status, or a map :



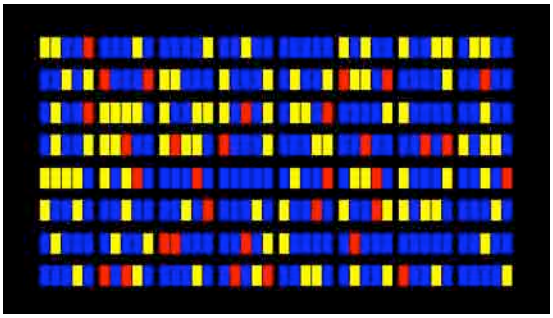
Another possibility is to group the 5 subparts for each antenna and keep a general matrix organization :



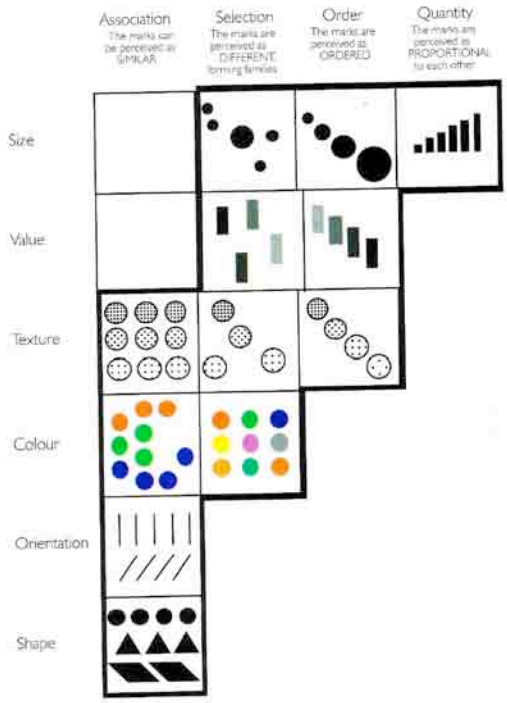
5.2.2 Value coding

The state of subsystem is described as 1) un-assigned 2) nominal 3) low priority alarm 4) high priority 5) extreme priority. [29, p. 16] suggests a color coding of rectangular boxes for these 5 levels : 1) blue 2) green 3) yellow 4) red 5) blinking red.

This can be used directly with the 3 spatial organizations defined in the last paragraph. For instance :



Other techniques could be investigated : variation of grain (grey level), variation of size, variation of shapes, etc. A classical study of these "retinal variables" is [3]. Bertin has shown that the various techniques are not equally good for all kind of graphics :



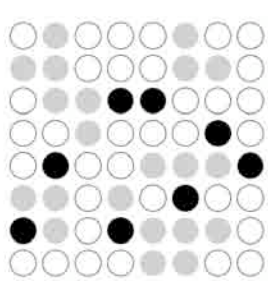
(from Spence "Information Visualisation")

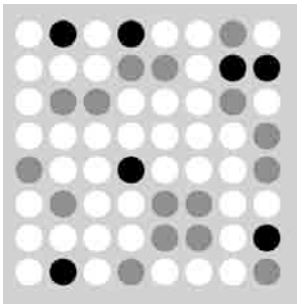
Variation of grey :

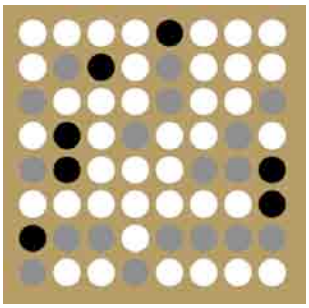
Here, we limit experimentations with 3 states only : nominal, low and high priority. A simplified mockup, using grey levels with rectangular shapes and systematic antenna numbering, can be derived from the actual chessboard. It suffers bad data/ink ratio because of shapes boundaries :

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

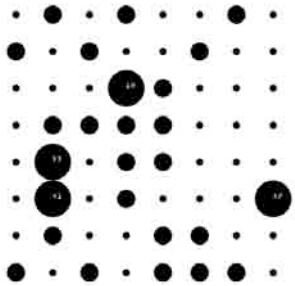
Using a circle for individual status possibly lead to a compact view. Using a grey background (grey200) enhances contrast. A color background can lead to perception problems ?



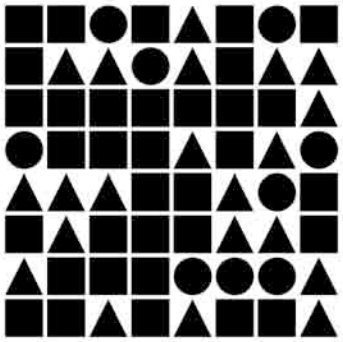


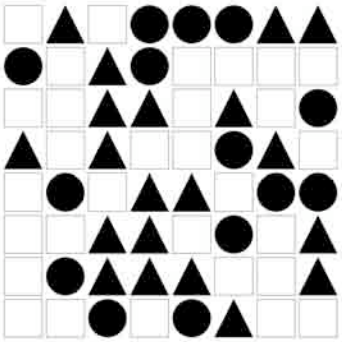


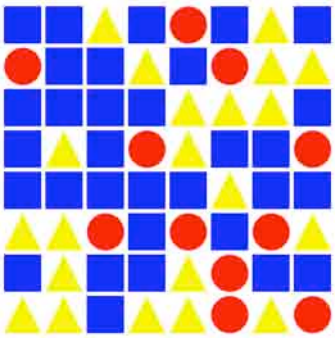
Variation of size can also be investigated. We assign larger size to higher priority alarm. Here, only antennas with hight priority alarms are numbered. It seems particularly efficient but is prone to change blindness. Progressive animation should be used.



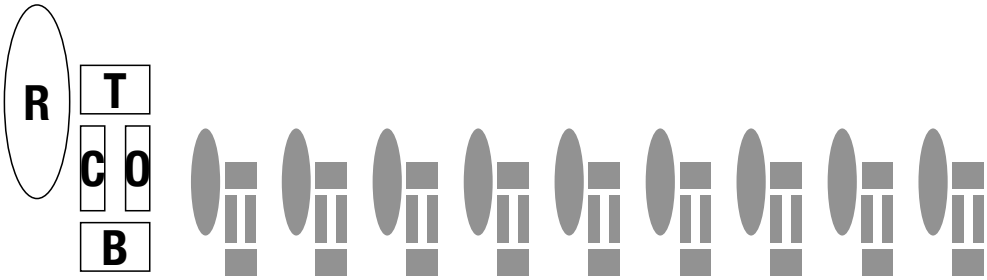
Variation of shape can be studied in a number of ways. One can assign specific shapes for the priority levels. But the association rules between shape and priority must be obvious (ie. very widely shared). Road signs are a good exemple : triangle for warning, circle for forbidden things, square for information can be associated to low, high and nominal respectively. Shapes can be associated with colors : red-circle, yellow-triangle, blue-square (à la Kandinsky). However, none of these schemes are particularly efficient here :



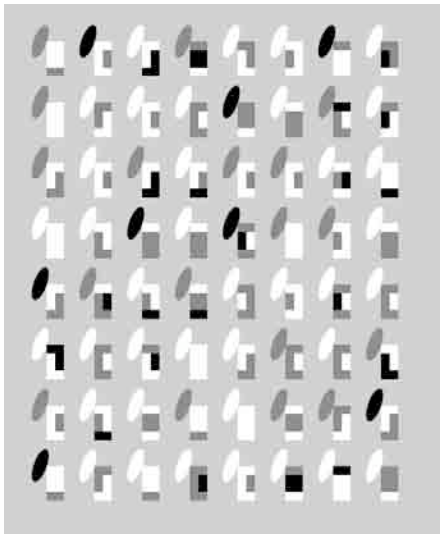




One can also rely on a simplified diagramatic view of an antenna and use it repeatedly to depict the full array status.

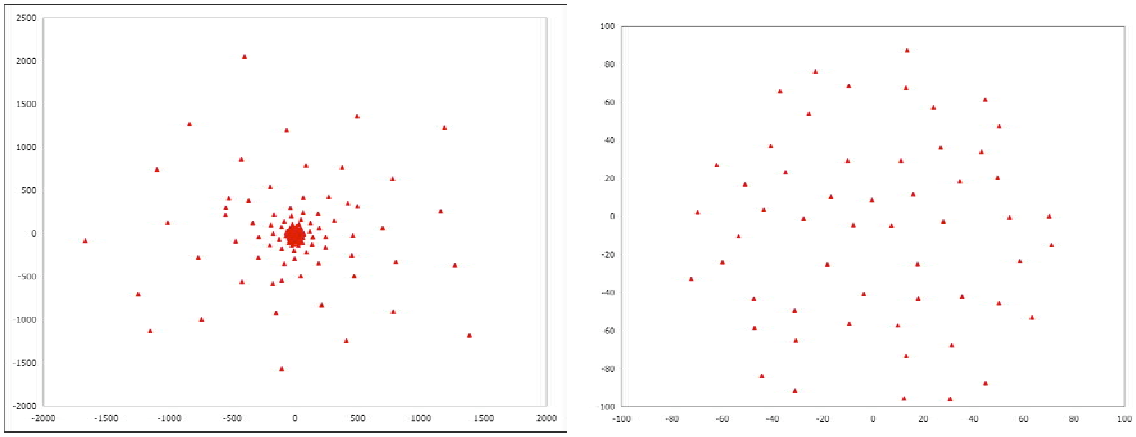


When a matrix organization is used, this may lead to readability problems, but the sequential organization might be studied further



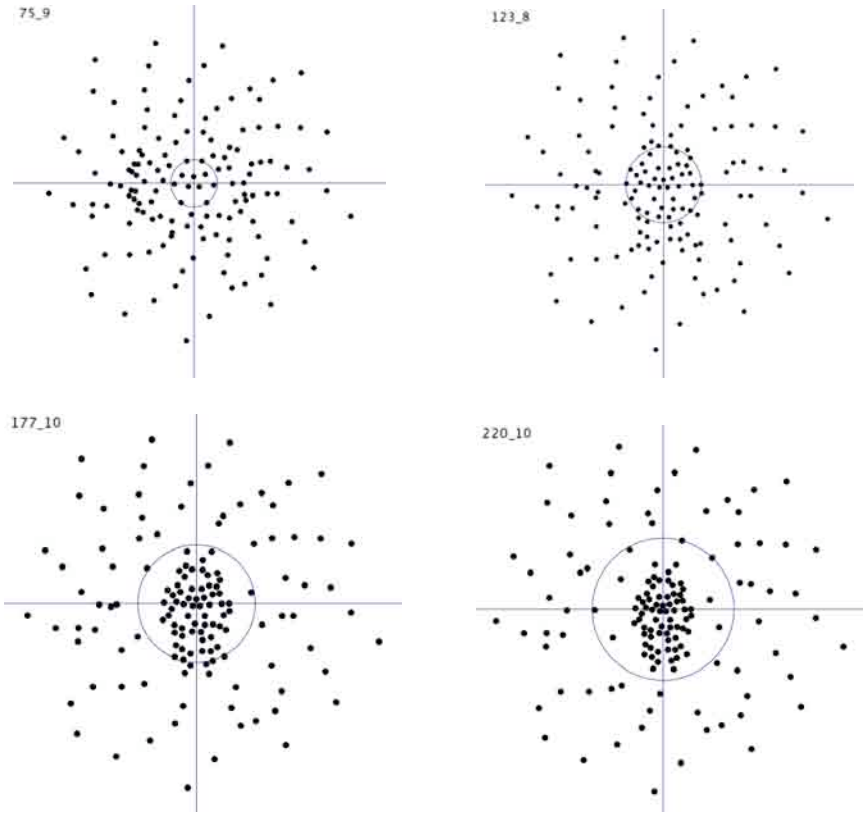
5.2.3 Cartographic approach

Antennas physical locations are pre-defined, among approx. 130 slots. The figure below depicts the location of these slots. A great dispersion happens but a spiral-like pattern is visible (note : if this pattern has some kind of importance in the general operation of ALMA, this should be taken into account in some way). Roughly half of the slots are located in the central area :

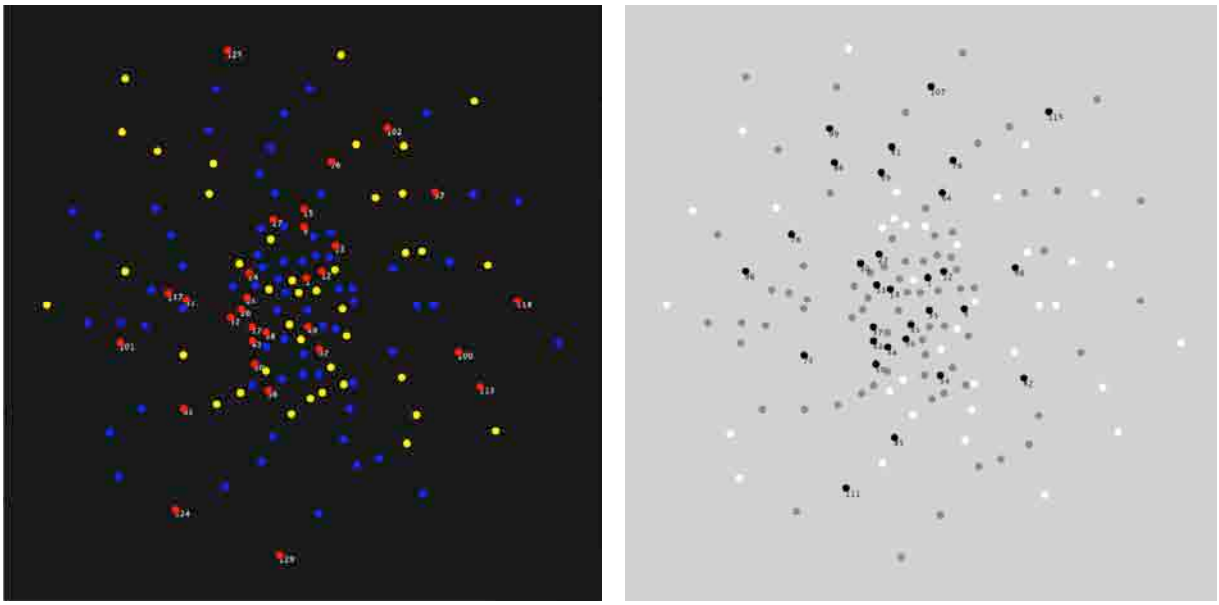


In order to provide a usefull synoptical view, two solutions are available : 1) offer two independant views : a global one and a detailled view for the center 2) use of a non-linear mapping.

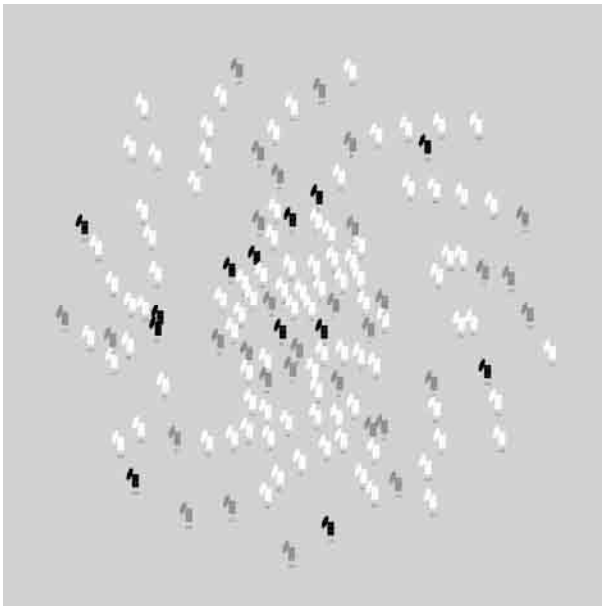
We investigate here the later. It seems difficult to use a standard logarithmic scale for both axes. A good solution is to translate rect. coordinates to polar, keep angle unchanged and scale distance d non linearly above some threshold D. In the figures below, the mapping for distance d is f(d) = D*(d/D)^0.33 for d > D and f(d) = d otherwise. Object size (in pixels) also varies (the bigger, the better). A good compromise for readability seems D=125, allowing an object size of 10 pix. (for a general picture being 600 pix. large)

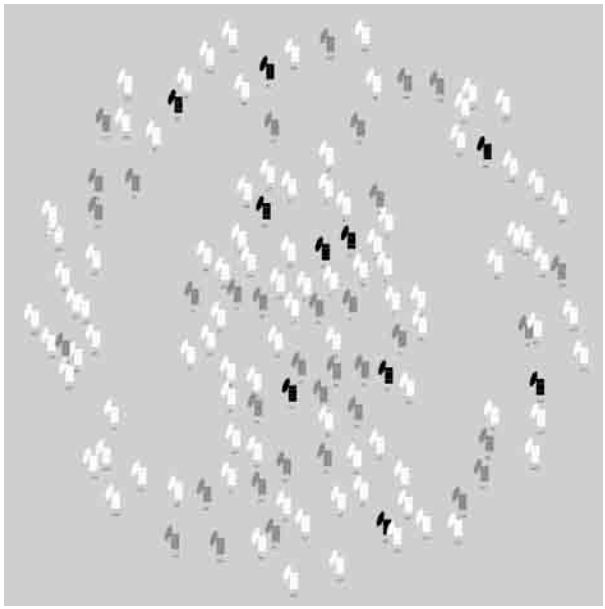


The status of the antennas can be rendered with all the techniques explored in the previous paragraph. For instance, using colors or grey values gives :



If we want to use a schematic antenna, the contraction exponent of the mapping formula needs to be decreased in order to scatter the points and increase the graphical objects dimensions. In doing so, the original organisation of the antennas array is not so easily recognized. (left : 1/5, right : 1/10)

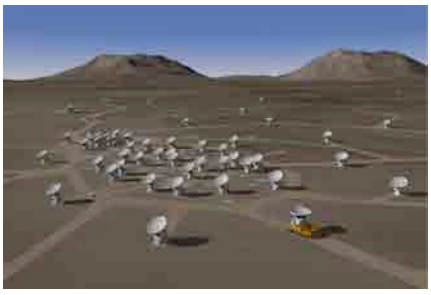




We must take into account the fact that not all slots can be used together (approximatly only half of them) so maybe this could lead to clearer, easier to read, maps. This needs further investigation with real data.

5.2.4. Realistic (3D) approach

Many photo-realistic simulations are provided by the ALMA project : posters for communication purposes, pictures on ALMA and ESO website. (it would be interesting to get in touch with the people who produced these pictures). Interesting examples are given below :



A first interest would be to show the pointing direction of the antennas (this is maybe naive : we are not sure of it usefullness). It could appear redundant with the video monitorings (at day light) but may be easily extended to include status informations, and, maybe, information flow between the arrays and correlator, etc. The main issue is that it will be difficult to cover the all area in one view, as it is the case for the standard 2D map.

References

[1] B. B. Bederson, J. Grosjean, and J. Meyer. Toolkit design for interactive structured graphics. *IEEE Trans. Softw. Eng.*, 30(8):535–546, 2004.

[2] B. B. Bederson, B. Shneiderman, and M. Wattenberg. Ordered and quantum treemaps: Making effective use of 2d space to display hierarchies. *ACM Trans. Graph.*, 21(4):833–854, 2002. http://doi.acm.org/10.1145/571647.571649.

[3] J. Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983. ISBN 0-299-09060-4.

[4] A. Caproni. ACS Alarm System, November 2007. ALMA.

[5] O. Chapuis, J.-B. Labrune, and E. Pietriga. Dynaspot: speed-dependent area cursor. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1391–1400, New York, NY, USA, 2009. ACM. http://doi.acm.org/10.1145/1518701.1518911.

[6] A. Cockburn, A. Karlson, and B. B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Computing Surveys*, 41(1):1–31, 2008. http://doi.acm.org/10.1145/1456650.1456652.

[7] Definition. WIMP (computing). http://en.wikipedia.org/wiki/WIMP_(computing).

[8] J.-D. Fekete and C. Plaisant. Excentric labeling: dynamic neighborhood labeling for data visualization. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 512–519, New York, NY, USA, 1999. ACM. http://doi.acm.org/10.1145/302979.303148.

[9] G. W. Furnas and B. B. Bederson. Space-scale diagrams: understanding multiscale interfaces. In *CHI '95: Proc. Human Factors in Computing Systems*, pages 234–241. ACM Press/Addison-Wesley, 1995. http://doi.acm.org/10.1145/223904.223934.

[10] P. Grosbøl, S. Scott, and M. Schilling. Executive Subsystem - Operator User Manual, November 2009. COMP-70.30.00.00-006-F-MAN.

[11] T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 281–290, New York, NY, USA, 2005. ACM. http://doi.acm.org/10.1145/1054972.1055012.

[12] J. Grudin. Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 458–465, New York, NY, USA, 2001. ACM. http://doi.acm.org/10.1145/365024.365312.

[13] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004. http://doi.acm.org/10.1145/993176.993177.

[14] C. Holz and S. Feiner. Relaxed selection techniques for querying time-series graphs. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 213–222, New York, NY, USA, 2009. ACM. http://doi.acm.org/10.1145/1622176.1622217.

[15] Y. Honglin. User Guide to the Quicklook Calibration Monitor, December 2009. ALMA-7-0-0, ACS-8.0.1.

[16] K. Hornbæk, B. B. Bederson, and C. Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Trans. Comput.-Hum. Interact.*, 9(4):362–389, 2002. http://doi.acm.org/10.1145/586081.586086.

[17] D. R. Hutchings and J. Stasko. Consistency, multiple monitors, and multiple windows. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 211–214, New York, NY, USA, 2007. ACM. http://doi.acm.org/10.1145/1240624.1240658.

[18] W. Javed and N. Elmqvist. Stack zooming for multi-focus interaction in time-series data visualization. In *Proceedings of the IEEE Pacific Visualization Symposium 2010*, 2010.

[19] G. Kurtenbach and W. Buxton. The limits of expert performance using hierarchic marking menus. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 482–487, New York, NY, USA, 1993. ACM. http://doi.acm.org/10.1145/169059.169426.

[20] S. Lucero. Scheduling Subsystem - User Guide for the Create Array Panel, Scheduling Panel and Project/SB Search Panel, November 2007. ALMA.

[21] T. Moscovich, F. Chevalier, N. Henry, E. Pietriga, and J.-D. Fekete. Topology-aware navigation in large networks. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 2319–2328, New York, NY, USA, 2009. ACM. http://doi.acm.org/10.1145/1518701.1519056.

[22] C. North and B. Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *AVI '00: Proc. working conference on Advanced Visual Interfaces*, pages 128–135. ACM Press, 2000. http://doi.acm.org/10.1145/345513.345282.

[23] K. Perlin and D. Fox. Pad: an alternative approach to the computer interface. In *SIGGRAPH '93: Proc. Computer Graphics and Interactive Techniques*, pages 57–64. ACM Press, 1993.

[24] Piccolo. A Structured 2D Graphics Framework. http://www.cs.umd.edu/hcil/jazz/.

[25] E. Pietriga and C. Appert. Sigma lenses: focus-context transitions combining space, time and translucence. In *CHI '08: Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems*, pages 1343–1352, New York, NY, USA, 2008. ACM. http://doi.acm.org/10.1145/1357054.1357264.

[26] E. Pietriga, O. Bau, and C. Appert. Representation-independent in-place magnification with sigma lenses. *IEEE Transactions on Visualization and Computer Graphics*, 16(1), 2010. http://doi.ieeecomputersociety.org/10.1109/TVCG.2009.98.

[27] A. Pucelj, I. Verstovsek, and A. Caproni. Logging Client User's Manual, April 2009. ALMA.

[28] S. Rankin and D. Hunter. CONTROL Device GUI User's Document. ALMA-7.0.0.

[29] S. Scott and D. Shepherd. Operations General User Interface (GUI) Requirements, November 2006. COMP-70.10.00.00-010-A-SPE.

[30] D. Shepherd, R. Laing, A. Peck, D. Emerson, J. Mangum, R. Lucas, T. Hunter, A. Hales, I. de Gregorio-Monsalvo, R. Aviles, and M. Olivare. GUI recommendations - Operator Interface for ALMA operations at the ATF and OSF, June 2007.

[31] B. Shneiderman. Direct manipulation: A step beyond programming languages. *Human-computer interaction: a multidisciplinary approach*, pages 461–467, 1987.

[32] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, 1994. http://dx.doi.org/10.1109/52.329404.

[33] B. Shneiderman, C. Plaisant, M. Cohen, and S. Jacobs. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (5th Edition)*. Addison Wesley, 2009. ISBN 0-321-53735-1.

[34] E. R. Tufte. *Visual Explanations*. Graphics Press, 1997. ISBN 0-9613921-2-6.

[35] ZVTM. Zoomable Visual Transformation Machine. http://zvtm.sourceforge.net.